CICS Transaction Server para z/OS
Versión 4 Release 2

Aplicaciones Java en CICS

SC11-7928-01

CICS Transaction Server para z/OS
Versión 4 Release 2

Aplicaciones Java en CICS

SC11-7928-01

Nota

Antes de utilizar esta información y el producto para el que brinda ayuda, lea la información que aparece en "Avisos" en la página 425.

Esta edición se aplica a Versión 4 Release 2 de CICS Transaction Server for z/OS (número de producto 5655-S97) y a todos los releases y las modificaciones posteriores, hasta que se indique algo distinto en nuevas ediciones.

Contenido

Prefacio

Este manual documenta interfaces de programación que permiten al cliente escribir programas para obtener los servicios de Versión 4 Release 2.

De qué trata esta información

Esta información le explica cómo desarrollar y utilizar aplicaciones Java y enterprise beans en CICS.

Quién debe leer esta información

Esta información está dirigida a:

- Programadores de aplicaciones Java experimentados que puedan tener poca experiencia con CICS y que no tengan gran necesidad de saber más sobre CICS de lo necesario para desarrollar y ejecutar programas Java.
- Usuarios y programadores del sistema CICS experimentados que deban tener conocimiento de los requisitos de CICS para el soporte de Java.

Cambios en CICS Transaction Server para z/OS, Versión 4 Release 2

Para obtener información sobre los cambios que se han efectuado en este release, consulte *Novedades* en el Information Center, o las siguientes publicaciones:

- *Novedades de CICS Transaction Server para z/OS*
- *Actualización de CICS Transaction Server para z/OS desde CICS TS versión 4.1*
- *Actualización de CICS Transaction Server para z/OS desde CICS TS versión 3.2*
- *Actualización de CICS Transaction Server para z/OS desde CICS TS versión 3.1*

Cualquier cambio técnico que se haga al texto después de la publicación se indica mediante una barra vertical (|) situada en la parte izquierda de cada nueva línea nueva o en la que se haya cambiado información.

Capítulo 1. Soporte de Java en CICS

CICS proporciona las herramientas y el entorno de ejecución para desarrollar y ejecutar aplicaciones empresariales Java en una máquina virtual Java (JVM) que esté bajo el control de una región CICS . Las aplicaciones Java pueden interaccionar con servicios y aplicaciones de CICS escritas en otro lenguaje.

Java en z/OS proporciona un soporte muy amplio para ejecutar aplicaciones Java. CICS utiliza el IBM® 64 bits SDK para z/OS, Java Technology Edition, versión 6.0.1. El SDK contiene un Java Runtime Environment que soporta el conjunto completo de API de Java y un conjunto de herramientas de desarrollo. Para fomentar la adopción de Java en z/OS, hay disponible un procesador especial en algunos hardware de System z. Este procesador se llama IBM System z Application Assist Processor (zAAP) y puede proporcionar capacidad de procesador adicional para ejecutar cargas de trabajo de Java elegibles con un coste reducido. CICS puede aprovechar esta prestación en sus cargas de trabajo de Java. Puede encontrar más información sobre Java en la plataforma z/OS, y descargar la versión de 64 bits del SDK en http://www.ibm.com/servers/eserver/zseries/software/java/.

CICS proporciona una herramienta basada en Eclipse y dos entornos de ejecución para aplicaciones Java:

CICS Explorer SDK

CICS Explorer SDK puede descargarse de forma gratuita para entornos de desarrollo integrado (IDE) basados en Java. El SDK proporciona soporte para el desarrollo y el despliegue de aplicaciones que cumplen la especificación de la Plataforma de servicios OSGi. La Plataforma de servicios OSGi proporciona un mecanismo para desarrollar aplicaciones utilizando un modelo de componentes y desplegar dichas aplicaciones en una infraestructura como paquetes OSGi. Un *paquete OSGi* es la unidad de despliegue para un componente de la aplicación y contiene información de control de versiones, dependencias y código de aplicación. El principal beneficio de OSGi es que se pueden crear aplicaciones a partir de componentes reutilizables a los que se accede únicamente mediante interfaces bien definidas denominadas *servicios OSGi*. También se pueden gestionar el ciclo de vida y las dependencias de aplicaciones Java de un modo granular.

El CICS Explorer SDK soporta el desarrollo de aplicaciones Java para cualquier versión soportada de CICS. El SDK incluye la biblioteca de clases Java CICS (JCICS) para acceder a servicios de CICS y ejemplos para iniciarse en el desarrollo de aplicaciones para CICS. También puede utilizar la herramienta para convertir aplicaciones Java existentes para OSGi.

Servidor de JVM

El servidor de JVM es el entorno de ejecución estratégico para aplicaciones Java en CICS. Un servidor de JVM puede manejar varias solicitudes simultáneas de distintas aplicaciones Java en una única JVM. Esto reduce el número de JVM que son necesarias para ejecutar aplicaciones Java en una región CICS. Para utilizar un servidor de JVM, las aplicaciones Java deben ser de enhebramiento seguro y deben cumplir con la especificación OSGi. Utilice este entorno de ejecución para todas las aplicaciones Java

donde sea posible. Este es el método preferido para ejecutar cargas de trabajo de Java en una región CICS y proporciona los siguientes beneficios:

- Puede ejecutar más de una aplicación Java en un servidor de JVM, lo que simplifica las operaciones de ejecutar y gestionar JVM en una región CICS.
- También puede ejecutar cargas de trabajo Java elegibles en zAAPs, reduciendo el coste de las transacciones.
- Puede ejecutar distintos tipos de trabajo en un servidor de JVM, incluidos programas Java de enhebramiento seguro y servicios web.
- Puede gestionar el ciclo de vida de aplicaciones en la infraestructura OSGi sin tener que reiniciar el servidor de JVM.
- Puede transportar más fácilmente aplicaciones Java que se hayan empaquetado utilizando OSGi entre CICS y otras plataformas.

JVM en agrupación

La JVM en agrupación es un entorno de ejecución en el que cada programa Java utiliza su propia JVM. Los programas de JVM que se ejecutan de forma simultánea están aislados entre sí. Cuando un programa Java ha terminado de utilizar la JVM, esta puede ser reutilizada por un programa subsiguiente. Utilice este entorno de ejecución para aplicaciones Java existentes que no sean de enhebramiento seguro. Las JVM en agrupación son estables y se eliminarán en una versión futura de CICS. Si es posible, convierta sus aplicaciones Java existentes para que se ejecuten en un servidor de JVM.

La plataforma de servicios OSGi

La plataforma de servicios OSGi proporciona un mecanismo para desarrollar aplicaciones utilizando un modelo de componentes y desplegar dichas aplicaciones en una infraestructura OSGi. La arquitectura OSGi está separada en varias capas que proporcionan beneficios para crear y gestionar aplicaciones Java.

La infraestructura OSGi es el núcleo de la especificación de la Plataforma de servicios OSGi. CICS utiliza la implementación Equinox versión 3.6.1 de la infraestructura OSGi, que admite la versión 4 de la especificación de la Plataforma de servicios OSGi. La infraestructura OSGi se inicializa cuando se lanza un servidor de JVM. Usar OSGi para aplicaciones Java brinda los siguientes beneficios principales:

- Las aplicaciones Java son más portátiles, más fáciles de rediseñar y más adaptables a los cambios en los requisitos.
- Puede seguir el modelo de programación Plain Old Java Object (POJO), que le ofrece la opción de desplegar una aplicación como un conjunto de paquetes OSGi con ciclos de vida dinámicos.
- Puede gestionar y administrar más fácilmente dependencias y versiones de paquete de aplicación.

La arquitectura OSGi tiene las siguientes capas:

- Capa de módulos
- Capa de ciclo de vida
- Capa de servicios

Capa de módulos

La unidad de despliegue es un paquete OSGi. La capa de módulos es donde la infraestructura OSGi procesa los aspectos modulares de un paquete. Los metadatos que permiten a la infraestructura OSGi realizar este proceso se proporcionan en un archivo manifiesto de paquete.

Una ventaja clave de OSGi es su modelo de cargador de clases, que utiliza los metadatos del archivo manifiesto. En OSGi no hay ninguna vía de acceso de clases global. Cuando los paquetes se instalan en la infraestructura OSGi, sus metadatos los procesa la capa de módulos y sus dependencias externas declaradas se reconcilian según las exportaciones y la información sobre la versión declaradas por otros módulos instalados. La infraestructura OSGi resuelve todas las dependencias y calcula la vía de acceso de clases necesaria e independiente para cada paquete. Este método resuelve los problemas de la carga de la clase Java simple al asegurar que los siguientes requisitos se cumplen:

- Cada paquete proporciona visibilidad únicamente a los paquetes Java que exporta de forma explícita.
- Cada paquete declara sus dependencias de paquete de forma explícita.
- Los paquetes se pueden exportar en versiones específicas e importarse a versiones específicas o desde un rango específico de versiones.
- Puede haber varias versiones de un paquete disponibles a la vez para clientes distintos.

Capa de ciclo de vida

La capa de gestión del ciclo de vida en OSGi permite que los paquetes se instalen, se lancen, se detengan y se desinstalen de forma dinámica, independientemente del ciclo de vida de la máquina virtual Java. La capa de ciclo de vida garantiza que los paquetes se inicien únicamente si sus dependencias están resueltas, lo que reduce el número de excepciones ClassNotFoundException en tiempo de ejecución. Si hay dependencias sin resolver, la infraestructura OSGi informa del problema y no lanza el paquete.

Cada paquete puede proporcionar una clase de activador de paquete, que se identifica en el manifiesto de paquete y que la infraestructura llama para iniciar y detener sucesos.

Capa de servicios

La capa de servicios de OSGi admite intrínsecamente una arquitectura orientada a servicios mediante su componente de registro de servicios no duraderos. Los paquetes publican servicios en el registro de servicio y otros paquetes pueden descubrir estos servicios desde el registro de servicio. Estos servicios son los medios primarios de colaboración entre paquetes. Un servicio OSGi es un Plain Old Java Object (POJO), publicado en el registro de servicio bajo uno o más nombres de interfaz, con metadatos opcionales almacenados como propiedades personalizadas (pares de nombre/valor). Un paquete descubridor puede buscar un servicio en el registro de servicio por un nombre de interfaz y, potencialmente, puede filtrar los servicios que se buscan en función de las propiedades personalizadas.

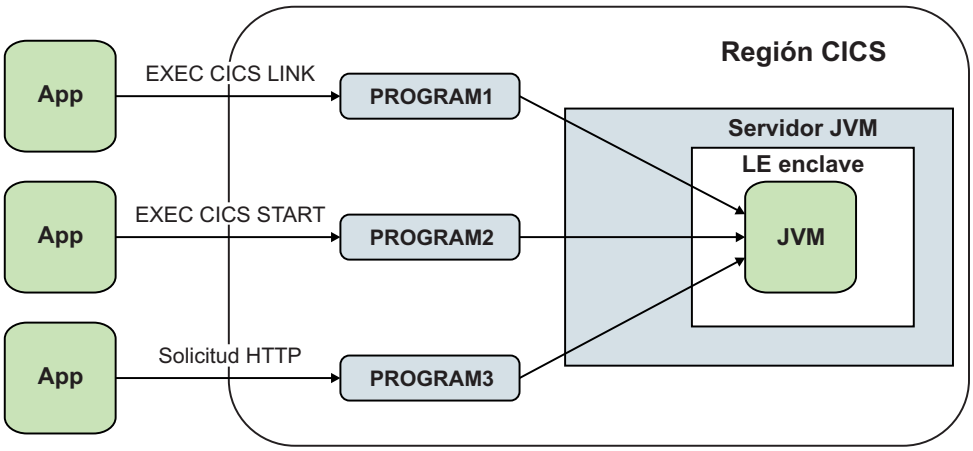
Los servicios son completamente dinámicos y, generalmente, tienen el mismo ciclo de vida que el paquete que los proporciona.

Entorno en tiempo de ejecución del servidor de JVM

Un *servidor de JVM* es un entorno de ejecución que puede gestionar varias solicitudes simultáneas para distintas aplicaciones Java en una única JVM de 64 bits. Puede utilizar un servidor de JVM para ejecutar aplicaciones Java de enhebramiento seguro en una infraestructura OSGi y procesar solicitudes de servicio web en el motor de servicios web Axis2.

Un servidor de JVM se representa mediante el recurso JVMSERVER. Cuando se habilita un recurso JVMSERVER, CICS solicita almacenamiento a MVS, configura un enclave de Language Environment, y lanza la JVM de 64 bits en el enclave. CICS utiliza un perfil de JVM que se especifica en el recurso JVMSERVER para crear la JVM con las opciones correctas. En este perfil se pueden añadir bibliotecas nativas para acceder a WebSphere MQ desde aplicaciones Java y especificar opciones de JVM. Java en z/OS gestiona de forma eficiente la memoria y la recogida de basura de la JVM, por lo que no es necesario definir estas opciones en el perfil.

Una de las ventajas de utilizar servidores de JVM es que se pueden ejecutar varias solicitudes de distintas aplicaciones en la misma JVM. En el siguiente diagrama, tres aplicaciones llaman a tres programas Java en una región CICS simultáneamente utilizando métodos de acceso distintos. Cada programa Java se ejecuta en el mismo servidor de JVM.



Aplicaciones Java

Para ejecutar una aplicación Java en un servidor de JVM, esta debe ser de enhebramiento seguro y estar empaquetada como uno o más paquetes OSGi en un paquete de CICS. El servidor de JVM implementa una infraestructura OSGi en la que se pueden ejecutar paquetes y servicios de OSGi. La infraestructura OSGi registra los servicios y gestiona las dependencias y las versiones entre los paquetes. OSGi maneja toda la gestión de vía de acceso de clases de la infraestructura por lo que puede añadir, actualizar y eliminar aplicaciones Java sin detener y reiniciar el servidor de JVM.

La unidad de despliegue para una aplicación Java que se empaquete utilizando OSGi es un paquete de CICS. Un paquete de CICS debe estar disponible en un directorio de zFS que contenga los paquetes OSGi. El recurso BUNDLE representa la aplicación para CICS y se puede utilizar para gestionar el ciclo de vida de la aplicación. El CICS Explorer SDK proporciona soporte para desplegar paquetes OSGi en un proyecto de paquete CICS en zFS.

Para acceder a la aplicación Java desde fuera de la infraestructura OSGi, utilice un recurso PROGRAM para identificar el servidor de JVM en el que se ejecuta dicha aplicación y el nombre del servicio OSGi. El servicio OSGi apunta a la clase principal de CICS.

Para obtener más información sobre el uso de de la infraestructura OSGi en un servidor de JVM, consulte "Aplicaciones Java compatibles con OSGi" en la página 24.

Servicios web

Puede utilizar un servidor de JVM para ejecutar el proceso SOAP de un solicitante de servicio web y aplicaciones de proveedor. Si una interconexión utiliza Axis2 2 (un motor SOAP basado en Java), el proceso SOAP se produce en un servidor de JVM. La ventaja de utilizar un servidor de JVM para servicios web es que puede descargar el trabajo a un procesador zAAP.

Para obtener más información sobre el uso de un servidor de JVM para servicios web, consulte "Servicios web de Java" en la página 20.

TCB TP y T8

CICS utiliza el entorno de transacciones abiertas (OTE) para ejecutar el trabajo del servidor de JVM. Cada tarea se ejecuta como una hebra en el servidor de JVM y se adjunta utilizando un TCB T8. El servidor de JVM también tiene un TCB padre llamado TP. El TCB TP se crea cuando se inicializa el servidor de JVM y se ejecuta en una hebra del sistema. La hebra del sistema proporciona acceso para consultar el estado del servidor de JVM, recopilar información de estadísticas y detener el servidor de JVM.

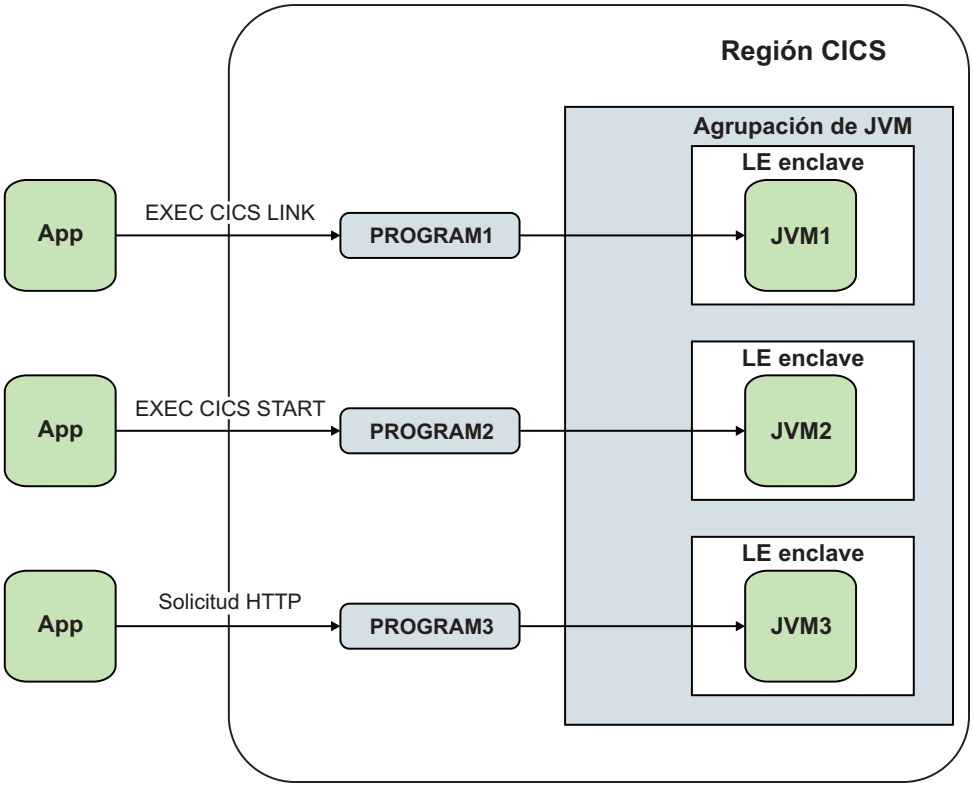
Cada tarea se adjunta a una hebra de la JVM utilizando un TCB T8. Puede controlar cuántos TCB T8 están disponibles para el servidor de JVM definiendo el atributo THREADLIMIT en el recurso JVMSERVER. Los TCB T8 que se crean para el servidor de JVM existen en una agrupación virtual y no puede reutilizarlos otro servidor de JVM que se ejecute en la misma región CICS. El número máximo de TCB T8 que puede existir en una región CICS en todos los servidores de JVM es 1024, mientras que el máximo para un servidor de JVM concreto es 256.

JVM en agrupación

Una *JVM en agrupación* es una JVM que puede gestionar una sola solicitud en cada momento desde una tarea de CICS. La agrupación para estas JVM pueden gestionar varias tareas a la vez, lo que significa que debe tener muchas JVM para gestionar cargas de trabajo de Java. CICS utiliza el entorno de transacciones abiertas (OTE) para ejecutar JVM en agrupación, y se puede ejecutar el número de JVM que CICS pueda crear en la región.

Una JVM en agrupación ejecuta un único programa Java para asegurarse de que cada transacción que implique a la JVM esté aislada de cualquier otra transacción simultánea que implique a una JVM. Por lo tanto, debe tener varias JVM disponibles para gestionar programas Java de forma simultánea. Para todas las nuevas cargas de trabajo Java, utilice el entorno de ejecución del servidor de JVM. En un servidor de JVM, puede ejecutar varios programas Java a la vez utilizando una única JVM.

En el siguiente diagrama, tres aplicaciones llaman a tres programas Java en una región CICS simultáneamente utilizando métodos de acceso distintos. Cada solicitud debe ejecutarse en una JVM y en un enclave distintos.



Reutilización de JVM

Cuando un programa Java ha finalizado, una JVM en agrupación se puede volver a asignar a otro programa Java. El perfil de JVM determina las características de una JVM y si esta se puede reutilizar o no. Si una JVM se puede reutilizar, se denomina *JVM continua*. Si una JVM no se puede reutilizar, se denomina *JVM de uso único*. Si debe utilizar JVM en agrupación, utilice JVM continuas para mejorar el rendimiento. También puede utilizar la memoria caché de clase compartida con JVM continuas para reducir los requisitos de almacenamiento y mejorar el tiempo de inicio para las JVM.

Las JVM continuas se pueden reutilizar muchas veces. El código de aplicación que se ejecute en el siguiente programa Java o la siguiente transacción no se aísla automáticamente de las acciones de la invocación del programa anterior; es decir, el aislamiento en serie no es automático. Debe asegurarse de que sus programas de aplicación Java no cambian el estado de una JVM continua de formas no deseables, ni dejan cualquier estado sin desear en la JVM.

Una JVM continua mantiene el contenido de sus almacenamientos dinámicos entre una invocación de programa y la siguiente. El estado estático o dinámico de los almacenamientos dinámicos de JVM continuas, así como las hebras que no se desactivan temporalmente, persisten, junto con su almacenamiento relacionado. Todas las clases de aplicación que se han cargado en las JVM se mantienen intactas. La aplicación puede elegir limpiar cualquier elemento no deseado y retener cualquier elemento que se desee.

El recurso PROGRAM para el programa Java determina la clave de ejecución adecuada y el perfil de JVM para la JVM que utiliza el programa. Puede definir distintos perfiles de JVM que cumplan los requisitos de los programas Java.

Cuando CICS recibe una solicitud para ejecutar un programa Java, debe crear una JVM adecuada o asignar una JVM existente que no se esté utilizando actualmente. Para crear una JVM adecuada, CICS solicita almacenamiento a MVS, configura un enclave de Language Environment y lanza la JVM en el enclave. CICS utiliza el perfil de JVM especificado en el recurso PROGRAM para crear la JVM con las clases y opciones correctas.

Límite para JVM en la agrupación de JVM

Cada JVM en agrupación se ejecuta en un TCB de MVS, que se asigna desde una agrupación de TCB abiertos J8 y J9. Esta agrupación de TCB abiertos se denomina la agrupación de JVM. Las JVM pueden estar en una de dos claves de ejecución: clave de usuario o clave de CICS. Las JVM que están en clave de usuario se ejecutan en un TCB J9. Las JVM que están en clave de CICS se ejecutan en un TCB J8. Las estadísticas se recopilan por separado para cada una de las modalidades, para poder ver qué proporciones de cada modalidad están en la agrupación de JVM. El perfil de JVM y la clave de ejecución son independientes entre sí, por lo que dos JVM podrían tener el mismo perfil pero distintas claves de ejecución.

El número total de TCB que se pueden crear para JVM está limitado por el parámetro de inicialización del sistema MAXJVMTCBS. Este parámetro limita el número de JVM que se puede tener en la agrupación de JVM de la región CICS.

Cada JVM se ejecuta en su propio enclave de Language Environment y utiliza almacenamiento de MVS. Por esta razón, debe elegir un límite de **MAXJVMTCBS** para su región CICS que tenga en cuenta no solo el tiempo de procesador utilizado por las JVM, sino también la cantidad de almacenamiento MVS que utiliza cada JVM y el almacenamiento disponible para la región. Si define un límite de **MAXJVMTCBS** que sea demasiado alto, CICS puede intentar crear demasiadas JVM para el almacenamiento MVS disponible, lo que resulta en una restricción de dicho almacenamiento MVS.

Perfiles de JVM

Los perfiles de JVM son archivos de texto que contienen opciones del lanzador Java y propiedades del sistema, las cuales determinan las características de las JVM. Puede editar los perfiles de JVM con cualquier editor de texto estándar.

Un perfil de JVM lista las opciones que utiliza el lanzador de CICS para Java. Alguna de las opciones son específicas para CICS y otras son estándar para el entorno de ejecución de la JVM. Por ejemplo, el perfil de JVM controla el tamaño inicial del almacenamiento dinámico y cuánto se puede expandir. El perfil también puede definir los destinos de los mensajes y la salida de volcado que produce la JVM.

El perfil de JVM también especifica las vías de acceso de clases. Las vías de acceso de clases contienen los directorios en los que la JVM busca las clases y los recursos de aplicación que necesitan las aplicaciones.

Cuando CICS recibe una solicitud para ejecutar un programa de Java, se pasa el nombre del perfil de JVM al lanzador de Java. El programa de Java se ejecuta en

una JVM, creada utilizando las opciones del perfil de JVM y del archivo de propiedades de JVM, si se ha especificado una.

CICS utiliza perfiles de JVM que están en el directorio de servicio del sistema z/OS UNIX especificado por el parámetro de inicialización del sistema JVMPROFILEDIR. Este directorio debe tener los permisos adecuados para que CICS lea los perfiles de JVM.

Perfiles de JVM de ejemplo

CICS proporciona cuatro perfiles de JVM de ejemplo para ayudarle a configurar el entorno Java. Estos se personalizan durante el proceso de instalación de CICS. Estos archivos los utiliza CICS como predeterminados o para programas de sistema.

Puede copiar los ejemplos y personalizarlos para sus propias aplicaciones. Los perfiles de JVM de ejemplo proporcionados por CICS están en el directorio `/usr/lpp/cicsts/cicsts42/JVMProfiles` en z/OS UNIX. Copie los ejemplos del directorio de instalación al directorio que especificara en el parámetro de inicialización del sistema **JVMPROFILEDIR**. Los perfiles de JVM de ejemplo de la ubicación de instalación se sobrescriben si se aplica un APAR que incluya cambios para estos archivos. Para evitar perder las modificaciones, copie siempre los ejemplos en una ubicación distinta antes de añadir sus propias clases de aplicación o de cambiar cualquier opción.

Los perfiles de JVM de ejemplo incluyen el símbolo &JAVA_HOME para la parte variable del nombre del directorio de instalación correspondiente a Java. Durante la instalación de CICS, este símbolo se sustituye por su propio valor. La vía de acceso a biblioteca base y la vía de acceso de clases base para la JVM, que no son visibles en el perfil de JVM, se crean automáticamente utilizando estos directorios. El valor predeterminado es `java/` para el símbolo &JAVA_HOME.

La siguiente tabla resume las característica clave de cada perfil de JVM de ejemplo.

Tabla 1. Perfiles de JVM de ejemplo proporcionados por CICS

perfil de JVM	Características
DFHJVMAX	El perfil DFHJVMAX es el perfil de ejemplo proporcionado para un servidor de JVM Axis2. El perfil de JVM se especifica en el recurso JVMSERVER. CICS utiliza el perfil DFHJVMAX para inicializar el servidor de JVM.No especifique este perfil en los recursos PROGRAM para sus propias aplicaciones. En su lugar, especifique el nombre del recurso JVMSERVER en el recurso PROGRAM.
DFHOSGI	El perfil DFHOSGI es el perfil de ejemplo proporcionado para un servidor de JVM OSGi. El perfil de JVM se especifica en el recurso JVMSERVER. CICS utiliza el perfil DFHJVMAX para inicializar el servidor de JVM.No especifique este perfil en los recursos PROGRAM para sus propias aplicaciones. En su lugar, especifique el nombre del recurso JVMSERVER en el recurso PROGRAM.
DFHJVMPR	El perfil DFHJVMPR es el predeterminado para las JVM en agrupación si no se especifica ningún perfil de JVM en el recurso PROGRAM de un programa Java. Las JVM en agrupación creadas con el perfil DFHJVMPR utilizan la memoria caché de clase compartida porque el perfil especifica CLASSCACHE=YES.

Tabla 1. Perfiles de JVM de ejemplo proporcionados por CICS (continuación)

perfil de JVM	Características
DFHJVMCD (reservado para el uso de CICS)	Los programas de sistema proporcionados por CICS tienen su propio perfil de JVM, DFHJVMCD, para las JVM en agrupación. Los programas de sistema son independientes de cualquier cambio que se realice en el perfil de JVM predeterminado, DFHJVMPR. En concreto, el recurso PROGRAM para el programa predeterminado procesador de solicitudes, DFJIIRP, especifica DFHJVMCD. Las JVM en agrupación creadas con el perfil DFHJVMCD no utilizan la memoria caché de clase compartida porque el perfil especifica CLASSCACHE=NO. Puede cambiar el valor predeterminado.No especifique este perfil en los recursos PROGRAM que configure para sus propias aplicaciones Java. Sin embargo, debe asegurarse de que se configura correctamente para su región CICS. CICS utiliza DFHJVMCD para inicializar y terminar la memoria caché de clase compartida además de utilizarlo para programas de sistema suministrados por CICS.

Estructura de una JVM

Las JVM que se ejecutan en CICS utilizan un conjunto de clases y de vías de acceso de clases que se definen en perfiles de JVM y utilizan almacenamiento de 64 bits. Cada JVM se ejecuta en un enclave de Language Environment que se puede ajustar para aprovechar el almacenamiento MVS de forma más eficaz.

Para obtener más información sobre la versión 6.0.1 del IBM 64 bits SDK para z/OS, Java Technology Edition, consulte la *IBM 64-bit SDK for z/OS, Java Technology Edition, versión 6.0.1 SDK and Runtime Environment User Guide*. El documento se puede descargar desde www.ibm.com/servers/eserver/zseries/software/java/javaintr.html.

Clases y vías de acceso de clases en las JVM

Hay tres tipos de clases y de bibliotecas nativas utilizados por una JVM que se ejecute en CICS.

- El código de JVM de z/OS, que proporciona los servicios base en la JVM. Estas clases son *clases del sistema* y *clases de extensión estándar*, que colectivamente se conocen como *clases primordiales*.
- Archivos nativos de biblioteca de enlace dinámico (DLL) de C que utiliza la JVM. Estos archivos tienen la extensión `.so` en z/OS UNIX. Algunas bibliotecas son necesarias para que la JVM se ejecute, y otras bibliotecas nativas pueden cargarlas el código o los servicios de aplicación. Por ejemplo, las bibliotecas nativas adicionales pueden incluir los archivos DLL para utilizar los controladores JBDC de DB2.
- Las clases Java para las aplicaciones que se ejecutan en la JVM. Estas clases se conocen como *clases de aplicación*. Este grupo incluye clases que forman parte de aplicaciones escritas por el usuario. También incluye algunas clases proporcionadas por IBM o por otro proveedor para brindar servicios que acceden a recursos, como las clases de interfaz de JCICS, JDBC y JNDI, que no se incluyen en la configuración de JVM estándar para CICS. Cuando las clases de aplicación se han cargado, se mantienen en las reutilizaciones de JVM, de forma que puedan utilizarlas otras transacciones.

La JVM entiende la finalidad de cada uno de estos elementos y determina cómo la JVM carga la clase o la biblioteca nativa y dónde se almacena esta.

Las vías de acceso de clases para una JVM se definen mediante opciones en el perfil de JVM y, opcionalmente, están en los archivos de propiedades de JVM referenciados.

Las vías de acceso de clases o las bibliotecas nativas se pueden incluir del siguiente modo:

- La *vía de acceso a biblioteca* es para todos los archivos nativos de biblioteca de enlace dinámico (DLL) de C que utilizan la JVM, incluidos los archivos necesarios para ejecutar la JVM y las bibliotecas nativas adicionales cargadas por el código o por los servicios de una aplicación. Solo se carga una copia de cada archivo DLL, y todas las JVM lo comparten, pero cada JVM cuenta con su propia copia del área de datos estáticos para la DLL.

  La vía de acceso a biblioteca base para la JVM se crea automáticamente utilizando los directorios especificados por el parámetro de inicialización del sistema **USSHOME** y la opción **JAVA_HOME** en el perfil de JVM. La vía de acceso a biblioteca base no es visible en el perfil de JVM. Incluye todos los archivos DLL necesarios para ejecutar la JVM, así como las bibliotecas nativas que utiliza CICS. Puede ampliar la vía de acceso a biblioteca con la opción **LIBPATH_SUFFIX** o con la opción **LIBPATH_PREFIX**. **LIBPATH_SUFFIX** añade elementos al final de la vía de acceso a biblioteca, tras las bibliotecas proporcionadas por IBM. **LIBPATH_PREFIX** añade elementos al principio, que se cargan en lugar de las bibliotecas que proporciona IBM si tienen el mismo nombre. Tal vez deba hacer esto para la determinación de problemas.

  Compile y enlace con la opción de LP64 cualquier archivo DLL que incluya en la vía de acceso a biblioteca. Los archivos DLL proporcionados en la vía de acceso a biblioteca base y los archivos DLL utilizados por servicios como los controladores DB2 JDBC se compilan con la opción LP64.

- La *vía de acceso de clase estándar* es para todas las clases de aplicación que se ejecutan en JVM agrupadas o en un servidor de JVM que no está configurado para OSGi. Todos los archivos `.class` y `.jar` de Java se colocan en la vía de acceso de clase estándar. Puede añadir clases a la vía de acceso de clase estándar utilizando la opción **CLASSPATH_SUFFIX** en el perfil de JVM o la opción **CLASSPATH_PREFIX**.

  CICS también crea una vía de acceso de clases base para la JVM automáticamente, utilizando los subdirectorios `/lib` de los directorios especificados por el parámetro de inicialización del sistema **USSHOME**. Esta vía de acceso de clases contiene los archivos JAR proporcionados por CICS y por la JVM. No es visible en el perfil de JVM.

  En el caso de los servidores de JVM configurados para admitir OSGi, no debe definir una vía de acceso de clases para las clases de aplicación. La infraestructura OSGi determina de forma automática la vía de acceso de clases de las aplicaciones utilizando la información del paquete OSGi que contiene la aplicación.

No tiene que incluir las clases del sistema y las clases de extensión estándares (las clases primordiales) en una vía de acceso de clases, porque ya están incluidas en la vía de acceso de clases de arranque en la JVM.

Almacenamiento dinámico en las JVM

El almacenamiento en tiempo de ejecución en las JVM para IBM 64 bits SDK para z/OS, Java Technology Edition versión6.0.1 se gestiona mediante un único almacenamiento dinámico de 64 bits.

El almacenamiento dinámico para cada JVM se asigna desde un almacenamiento de 64 bits en el enclave de Language Environment para la JVM. El tamaño de cada almacenamiento dinámico se determina mediante opciones en el perfil de JVM.

El almacenamiento dinámico único se conoce como el *almacenamiento dinámico*, o, a veces, como el *almacenamiento dinámico de elementos no utilizados*. Su asignación de almacenamiento inicial se define mediante la opción **-Xms** en un perfil de JVM, y su tamaño máximo se define mediante la opción **-Xmx**.

Puede ajustar el tamaño de un almacenamiento dinámico para lograr un rendimiento óptimo para sus JVM. Consulte "Ajuste del almacenamiento dinámico y de la recogida de basura del servidor de JVM" en la página 170 y "Ajuste de los almacenamientos dinámicos y de la recogida de basura de JVM en agrupación" en la página 179.

Dónde se construyen las JVM

Si es necesaria una JVM, el programa lanzador de CICS para JVM solicita almacenamiento a MVS, configura un enclave de Language Environment y lanza la JVM en el enclave de Language Environment. Cada JVM se construye en su propio enclave de Language Environment para asegurar el aislamiento entre las JVM que se ejecutan en paralelo.

El enclave de Language Environment se crea utilizando el módulo de inicialización previa de Language Environment, CELQPIPI, y la JVM se ejecuta como un proceso de z/OS UNIX. Por tanto, la JVM utiliza servicios de Language Environment de MVS en lugar de servicios de Language Environment de CICS. El almacenamiento utilizado para una JVM es almacenamiento de 64 bits de MVS, obtenido mediante llamadas a servicios de Language Environment de MVS. Este almacenamiento reside en el espacio de direcciones de CICS, pero no se incluye en las área de almacenamiento dinámico (DSA) de CICS.

El enclave de Language Environment para una JVM se puede expandir, en función de los requisitos de almacenamiento de dicha JVM. Las opciones de tiempo de ejecución de Language Environment que utiliza CICS para un enclave de Language Environment controlan el tamaño inicial del almacenamiento dinámico del enclave de Language Environment y los añadidos incrementales al mismo.

Puede ajustar las opciones de tiempo de ejecución que CICS utiliza para un enclave de Language Environment, de forma que la cantidad de almacenamiento que CICS solicite para el enclave sea lo más aproximado posible a la cantidad de almacenamiento especificada por los perfiles de JVM. Por lo tanto, puede realizar el uso más eficiente del almacenamiento de MVS. Para obtener más información sobre el ajuste del almacenamiento, consulte "Almacenamiento del enclave de Language Environment para JVM" en la página 183.

Claves de ejecución para las JVM

Un programa Java debe utilizar una JVM que se ejecute en la clave de ejecución correcta. Las JVM en agrupación pueden ejecutarse en una de estas dos claves de ejecución: clave de usuario o clave de CICS. Los servidores de JVM se ejecutan solo en clave de CICS.

Claves de ejecución para servidores de JVM

Los servidores de JVM se ejecutan solo en clave de CICS. Para utilizar un servidor de JVM, el recurso PROGRAM para el programa debe tener el atributo EXECKEY

definido en CICS. CICS utiliza un TCB T8 para ejecutar la JVM y obtiene almacenamiento MVS en la clave de CICS.

Claves de ejecución para JVM en agrupación

Cuando se define el atributo EXECKEY en el recurso PROGRAM para un programa Java como USER, CICS concede al programa una JVM en agrupación que esté en clave de usuario. CICS utiliza un TCB J9 para ejecutar la JVM y obtiene almacenamiento MVS en la clave de usuario. Cuando se define el atributo EXECKEY como CICS, CICS concede al programa una JVM en agrupación que está en clave de CICS. CICS utiliza un TCB J8 para ejecutar la JVM y obtiene almacenamiento MVS en la clave de CICS.

La ejecución de aplicaciones en clave de usuario amplía la protección de almacenamiento de CICS, por lo que si los programas de Java utilizan una JVM en agrupación, se ejecutan en clave de usuario si es posible. Sin embargo, si el programa forma parte de una transacción que especifique **TASKDATAKEY**(CICS), dicho programa debe utilizar una JVM que se ejecute en clave de CICS.

No tiene que realizar ningún otro cambio si cambia el atributo EXECKEY para un recurso PROGRAM de Java. CICS puede utilizar el mismo perfil de JVM para crear JVM en ambas claves de ejecución. Una única tarea de CICS puede incluir programas de Java que se ejecuten en clave de CICS y programas Java que se ejecuten en clave de usuario. Sin embargo, una JVM la pueden reutilizar únicamente programas que especifiquen la misma clave de ejecución y el mismo perfil de JVM en los recursos PROGRAM. Si la mayoría de las JVM se crean en la misma clave de ejecución, CICS tiene más oportunidades para brindar a un programa una JVM existente para reutilizarla, en lugar de crear una nueva JVM.

JVM y la región de biblioteca compartida de z/OS

La región de biblioteca compartida es una característica de z/OS que permite que los espacios de direcciones compartan archivos de la biblioteca de enlaces dinámicos (DLL).

Esta función permite que las regiones de CICS compartan las DLL que necesiten las JVM, en lugar de que cada región tenga que cargarlas individualmente. Esto puede reducir enormemente la cantidad de almacenamiento real utilizado por MVS y el tiempo necesario para que las regiones carguen los archivos.

El almacenamiento reservado para la región de biblioteca compartida se asigna a cada región CICS cuando se lanza la primera JVM en la región. La cantidad de almacenamiento que se asigna se controla mediante el parámetro **SHRLIBRGNSIZE** en z/OS. Para obtener más información sobre cómo ajustar la cantidad de almacenamiento que se asigna para la región de biblioteca compartida, consulte “Ajuste de la región de biblioteca compartida de z/OS” en la página 191.

Memoria caché de clase compartida

El IBM SDK para z/OS proporciona un recurso de compartición de clases para JVM, donde varias JVM pueden compartir una misma memoria caché de archivos de clase que ya se han cargado. CICS soporta este recurso para JVM en agrupación y servidores de JVM de distintas maneras.

La memoria caché de clase compartida contiene todas las clases que necesitan las JVM que utilizan la memoria caché de clase compartida. Todas las clases de

aplicación requeridas por los programas de Java se colocan en la vía de acceso de clase estándar en los perfiles JVM y todas ellas son elegibles para cargarlas en la memoria caché de clase compartida. En algunos escenarios excepcionales, es posible que algunas clases no sean elegibles para su carga en la memoria caché de clase compartida.

La memoria caché de clase compartida no almacena los siguientes elementos.

- Archivos de biblioteca de enlace dinámico (DLL) de C que se especifican en la vía de acceso a biblioteca de los perfiles de JVM. Todas las JVM que la necesitan utilizan un única copia de cada archivo de DLL.
- Datos de trabajo para aplicaciones (objetos y variables). Los datos de trabajo se almacenan en las JVM individuales.
- Clases compiladas producidas mediante compilación Just In Time (JIT). Las clases compiladas se almacenan en JVM individuales, no en la memoria caché de clase compartida, porque el proceso de compilación puede variar para cargas de trabajo diferentes.

La memoria caché de clase compartida actualiza su contenido automáticamente si se cambia alguna clase de aplicación o algún archivo JAR, o si se añaden nuevos elementos a las vías de acceso de clase en los perfiles de JVM y se reinician las JVM adecuadas. La memoria caché de clase compartida es persistente en inicios de CICS en caliente o de emergencia, excepto en algunas circunstancias tales como un IPL de z/OS, por lo que no hay coste de inicio para la primera JVM de la región CICS en tales casos.

En Java 6.0.1, puede tener varias memorias caché de clase compartida disponibles para utilizar a la vez. CICS no proporciona interfaces para gestionar varias memorias caché de clase, pero puede utilizar varias memorias caché de clase compartida con servidores de JVM. Las JVM en agrupación no pueden utilizar varias memorias caché de clase, pero CICS sí brinda interfaces para gestionar una única memoria caché de clase en una región para JVM en agrupación.

Memoria caché de clase para servidores de JVM

Si desea utilizar memorias caché de clase con servidores de JVM, puede utilizar el soporte provisto por Java 6 directamente. Este soporte se describe en Class data sharing between JVMs. Los servidores de JVM no utilizan el soporte para memorias caché de clase que se proporciona en CICS. Por ejemplo, no puede habilitar ni inhabilitar una memoria caché de clase para servidores de JVM que utilizan mandatos de SPI o de CEMT.

Memoria caché de clase para JVM en agrupación

Las JVM en agrupación que utilizan la memoria caché de clase compartida se inician más rápido y tienen menos requisitos de objeto derivado que las JVM que no la utilizan. El coste total de la carga de clase también se reduce cuando las JVM en agrupación utilizan la memoria caché de clase compartida. Cuando se inicializa una nueva JVM que comparte la memoria caché de clase, utiliza las clases preinstaladas en lugar de leerlas del sistema de archivos. Una JVM que comparte la memoria caché de clase sigue siendo propietaria de los datos de trabajo (objetos y variables) para las aplicaciones que se ejecutan en ella, con el fin de mantener el aislamiento entre las aplicaciones Java que se procesan en el sistema.

CICS utiliza el perfil de ejemplo DFHJVMCD proporcionado por CICS para inicializar y terminar la memoria caché de clase compartida para JVM en

agrupación. DFHJVMCD debe estar siempre disponible y configurado para su uso en la región CICS, pero no es necesario realizar cambios adicionales para su uso con la memoria caché de clase compartida.

CICS proporciona interfaces para gestionar una memoria caché de clase compartida en cada región. Una región también puede contener memorias caché de clase compartida antiguas que se estén desactivando. Puede gestionar la memoria caché de clase compartida y supervisar su estado mediante mandatos de CICS.

La memoria caché de clase compartida recibe el nombre `CICS_sharedcc_`*`APPLID_n`*, donde *APPLID* es el identificador de aplicación de la región CICS y *n* es un número de generación que empieza en cero. El número de generación se utiliza para diferenciar el nombre de la nueva memoria caché de clase compartida.

CICS utiliza uno o más TCB JM, un tipo de TCB abierto, para funciones de gestión de memoria caché de clase compartida. Los TCB JM no cuentan para el límite de **MAXJVMTCBS** para la agrupación de JVM.

El parámetro de inicialización del sistema JVMCCSIZE especifica el tamaño inicial de la memoria caché de clase compartida. El parámetro de inicialización del sistema JVMCCSTART controla el comportamiento de inicio de la memoria caché de la clase compartida en la inicialización de la región CICS.

Capítulo 2. Planificación de Java

Si está planificando cómo utilizar Java en su empresa, los ejemplos de esta sección proporcionan orientación sobre las distintas opciones estratégicas que están disponibles para aplicaciones CICS.

Puede utilizar Java en CICS de distintos modos:

Utilice JCA para conectar aplicaciones Java externas a CICS.

Puede utilizar la Java EE Connector Architecture (JCA) para conectar aplicaciones CICS existentes con aplicaciones Java externas utilizando CICS Transaction Gateway. Este producto de la familia CICS proporciona soporte para conectar aplicaciones Java en servidores de aplicaciones, como WebSphere Application Server, con CICS utilizando adaptadores de recursos que implementen la tecnología JCA.

Las aplicaciones de CICS pueden escribirse en cualquiera de los lenguajes de programación de alto nivel soportados.

Utilización de servicios web de Java

Puede crear servicios web de Java para trabajar con proveedores de servicios y solicitantes de servicios en un entorno variado, conectándose a Internet mediante HTTP o WebSphere MQ. Los servicios web de Java se ejecutan en un servidor de JVM y el proceso SOAP lo realiza el motor de servicios web Axis2. Puede elegir procesar servicios web existente en Axis2, donde la aplicación del proveedor o del solicitante esté escrita en cualquiera de los lenguajes de programación de alto nivel soportados, incluido Java. También puede utilizar API estándar de Java para crear servicios web de Java que puedan manejar XML o trabajar con datos estructurados.

Las cargas de trabajo de Java que se ejecutan en un servidor de JVM son elegibles para ejecutarse en un procesador IBM System z Application Assist Processor (zAAP).

Utilización de OSGi para crear aplicaciones Java

Puede crear aplicaciones Java reutilizables y modulares que cumplan con la Plataforma de servicios OSGi. Estas aplicaciones son fáciles de mover entre CICS y otras plataformas, y OSGi brinda granularidad en la gestión de dependencias y versiones.

Puede utilizar las API de Java CICS (JCICS) para grabar aplicaciones que accedan a servicios de CICS, como leer desde archivos o colas de almacenamiento temporales. Las aplicaciones Java pueden enlazar con otras aplicaciones CICS y acceder a datos en DB2 e IMS. Las aplicaciones Java pueden ejecutarse en servidores de JVM o en JVM en agrupación. El entorno estratégico para la ejecución de aplicaciones Java es el servidor de JVM, así que planifique este entorno para todas las aplicaciones Java. Las cargas de trabajo de Java que se ejecutan en un servidor de JVM son elegibles para ejecutarse en un zAAP.

Como parte de su planificación, también debe decidir cómo direccionar las cargas de trabajo de Java y escalar sus regiones CICS de acuerdo con esto.

Acceso a aplicaciones CICS desde CICS Transaction Gateway

CICS Transaction Gateway proporciona adaptadores de recurso para conectar programas de cliente Java a aplicaciones CICS ya existentes.

Puede utilizar los adaptadores de recursos de CICS TG para reutilizar las aplicaciones CICS en nuevas aplicaciones Java. Frecuentemente, las nuevas aplicaciones Java se pueden desarrollar de forma más rápida y fiable reutilizando aplicaciones CICS ya existentes tanto Java como no. Lo normal es que la aplicación de cliente Java esté basada en red y que el programa de CICS esté escrito en un lenguaje como COBOL.

La J2EE Connector Architecture (JCA)

La Java 2 Platform Enterprise Edition (J2EE) Connector Architecture define una manera estándar de conectar una plataforma compatible con J2EE a un sistema de información empresarial (EIS) variado, como CICS. Las aplicaciones Java interaccionan con adaptadores de recursos utilizando la interfaz de cliente común (CCI), que es un estándar abierto definido por la JCA.

La J2EE Connector Architecture permite a un proveedor de EIS proporcionar un adaptador de recursos estándar para su EIS. Un adaptador de recursos es el nivel medio entre una aplicación Java y un EIS y permite a la aplicación Java conectarse con el EIS.

La CICS Transaction Gateway implementa el JCA al proporcionar adaptadores de recursos de CICS J2EE que soportan la interfaz de cliente común.

Acceso a programas de CICS desde programas Java externos

Desde la red, una aplicación de cliente Java puede utilizar cualquiera de los siguientes métodos para llamar a un programa de CICS TS:

La API de CICS TG
La API de CICS Transaction Gateway proporciona, entre otras cosas, los siguientes recursos:

La interfaz de llamada externa
Una aplicación externa puede utilizar la interfaz de llamada externa (ECI) para llamar a un programa en una región CICS. Para ser elegible, el programa de CICS debe ponerse a disposición de otros programas de CICS mediante un mandato **EXEC CICS LINK**. Puede tener una interfaz de COMMAREA o, cuando se utiliza una conexión IPIC, el programa puede utilizar un canal y contenedores para transferir datos.

Los programas de CICS llamados por una solicitud de ECI deben seguir las reglas para solicitudes de enlace de programa distribuido (DPL). Para obtener información sobre solicitudes de DPL, consulte Enlace de programa distribuido (DPL) en programaciones de aplicación CICS.

La interfaz de presentación externa
Una aplicación externa puede utilizar la interfaz de presentación externa (EPI) para llamar a un programa de aplicación CICS basado en 3270 y utilizar su salida. La aplicación cliente puede instalar y suprimir terminales IBM 3270 virtuales en la región CICS. CICS procesa las

definiciones utilizadas por la EPI como definiciones de terminal 3270 remoto y, por lo tanto, soporta las solicitudes de iniciación de transacción automática (ATI).

La interfaz de seguridad externa
Una aplicación externa puede utilizar la interfaz de seguridad externa (ESI) para realizar ciertas funciones de seguridad. Por ejemplo, la aplicación puede acceder a información sobre los ID de usuario que se mantienen en el gestor de seguridad externa (ESM) de CICS y definir las credenciales de seguridad predeterminadas para una conexión de servidor.

Los adaptadores de recursos ECI

Los adaptadores de recursos de la ECI proporcionan una interfaz CCI de alto nivel para la interfaz de llamada externa que las aplicaciones pueden utilizar para enlazar con aplicaciones CICS y pasar datos en COMMAREA o en contenedores. Los adaptadores de recursos se pueden desplegar en un servidor de aplicaciones J2EE, como WebSphere Application Server, para que las aplicaciones empresariales J2EE puedan acceder a CICS. Cuando se utiliza la JCA, la técnica de agrupación de conexiones, la seguridad y el contexto de transacciones las gestiona el servidor de aplicaciones de J2EE en lugar de la aplicación.

Para z/OS se proporcionan dos adaptadores de recursos ECI en CICS Transaction Gateway:

- El adaptador `cicseciXA.rar`, que admite confirmación en dos fases
- El adaptador `cicseci.rar`, que admite confirmación en una sola fase únicamente

Los adaptadores de recursos de la ECI también soportan las siguientes características adicionales:

Soporte para conexiones IPIC
Puede utilizar conexiones IPIC para acceder a CICS a través de TCP/IP si la región es CICS TS para z/OS, versión 3.2 o posterior. A diferencia de EXCI, APPC y ECI sobre TCP/IP, este tipo de conexión soporta contenedores y autenticación SSL. La conexión IPIC se representa mediante un recurso IPCONN en CICS.

No es posible instalar recursos IPCONN estáticos en clientes Java: dichas conexiones siempre se instalan de forma automática. Consulte Crear un programa para controlar la instalación automática de las conexiones IPIC en la Guía de personalización.

Canales y contenedores
Los canales y los contenedores brindan a las aplicaciones un modo de transferir datos en CICS que sean mayores de 32 KB. Para obtener más información sobre canales y contenedores, consulte Transferencia de datos entre programas mejorada utilizando canales en programación de aplicaciones CICS.

Autenticación de capa de sockets seguros (SSL)
Autenticación de capa de sockets seguros (SSL). La SSL se soporta en conexiones IPIC entre CICS Transaction Gateway y CICS. Para obtener información sobre la utilización de autenticación SSL, consulte Configuración de CICS para utilizar SSL en la Guía de seguridad RACF.

El adaptador de recursos EPI

El adaptador de recursos EPI proporciona una interfaz CCI de alto nivel para

la interfaz de presentación externa, que se puede utilizar para instalar terminales y ejecutar transacciones basadas en 3270 en una región CICS. No se soporta la iniciación de transacción automática (ATI). El adaptador de recursos se puede desplegar en un servidor de aplicaciones J2EE para que las aplicaciones empresariales J2EE accedan a CICS. Cuando se utiliza la JCA, la técnica de agrupación de conexiones, la seguridad y el contexto de transacciones las gestiona el servidor de aplicaciones de J2EE en lugar de la aplicación.

Ejemplos de utilización de adaptadores de recursos de CICS

El escenario que se muestra en Figura 1 en la página 19 es un ejemplo de una *configuración de 3 niveles*. Una aplicación Java utiliza el adaptador de recursos ECI para enlazar con un programa en la región CICS. La conexión entre la aplicación cliente y la región CICS se produce mediante un sistema intermedio. Debido a que la aplicación cliente no se ejecuta en el mismo host que CICS Transaction Gateway, el daemon escucha al cliente y se comunica con él. En z/OS, CICS Transaction Gateway utiliza la interfaz CICS externa (EXCI) o el controlador de IPIC para pasar solicitudes a CICS. Estas solicitudes las procesa CICS como llamadas de ECI.

El diagrama también muestra un servlet Java que también utiliza el adaptador de recursos ECI para conectar con un programa de servidor. Esta configuración es un ejemplo de una *configuración de dos niveles*, en la que existe una conexión directa entre la aplicación cliente y la región CICS mediante el adaptador ECI. Como el servlet se ejecuta en el mismo host que la CICS TG, utiliza el protocolo local para comunicarse.

CICS Transaction Gateway en z/OS admite la interfaz de llamada externa pero no la interfaz de presentación externa. Se soportan la ECI y el adaptador de recursos ECI, pero no la EPI ni el adaptador de recursos EPI. Solo se soportan los programas de cliente Java. Las llamadas a la ECI pueden efectuarse mediante conexiones EXCI o IPIC con la región CICS.

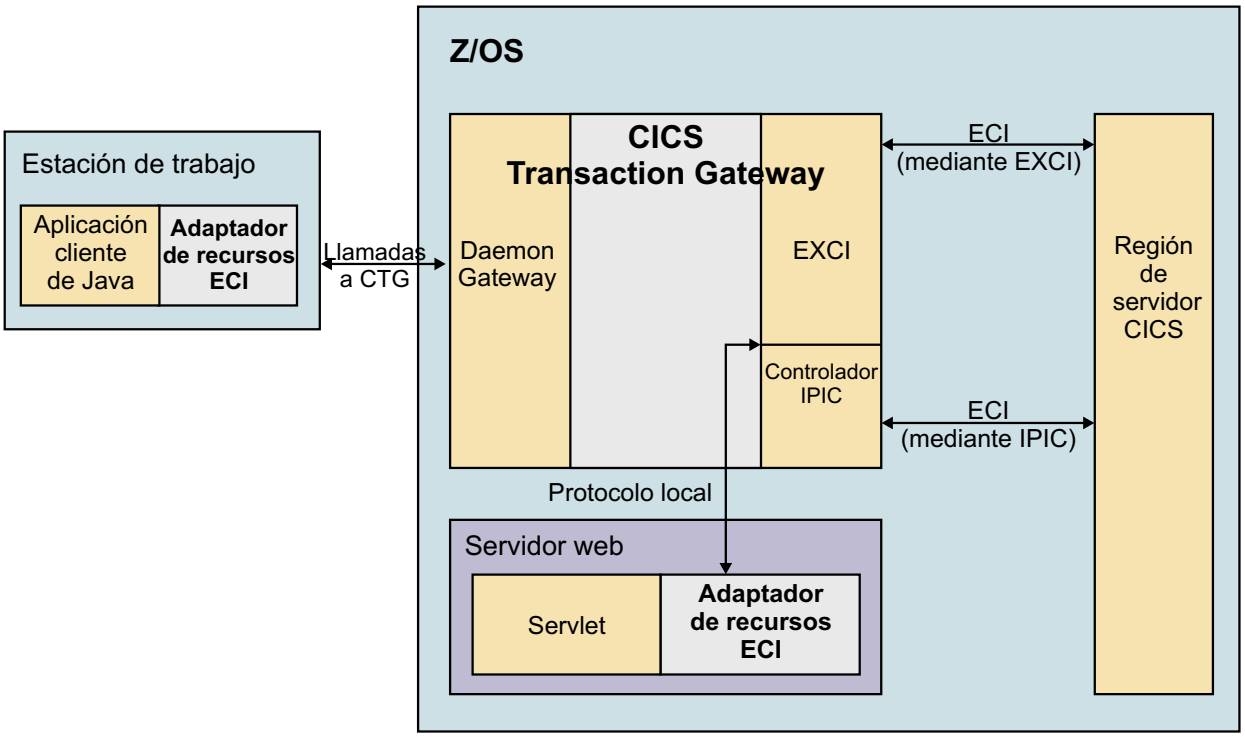


Figura 1. Los clientes Java se conectan con un programa de CICS utilizando el adaptador de recursos ECI

Se muestra una variación en la Figura 2 en la página 20. En este ejemplo, CICS Transaction Gateway se ejecuta en un servidor Windows. La CICS Transaction Gateway en Windows y Linux soporta la ECI y el adaptador de recursos ECI y la EPI y el adaptador de recursos EPI. El cliente Java puede acceder a programas de CICS basados en 3270 así como a programas de CICS que utilicen una COMMAREA o contenedores adecuados.

Las llamadas de ECI pueden efectuarse a CICS sobre APPC, TCP62, ECI sobre TCP/IP o conexiones IPIC. Las llamadas de EPI solo se soportan en conexiones APPC.

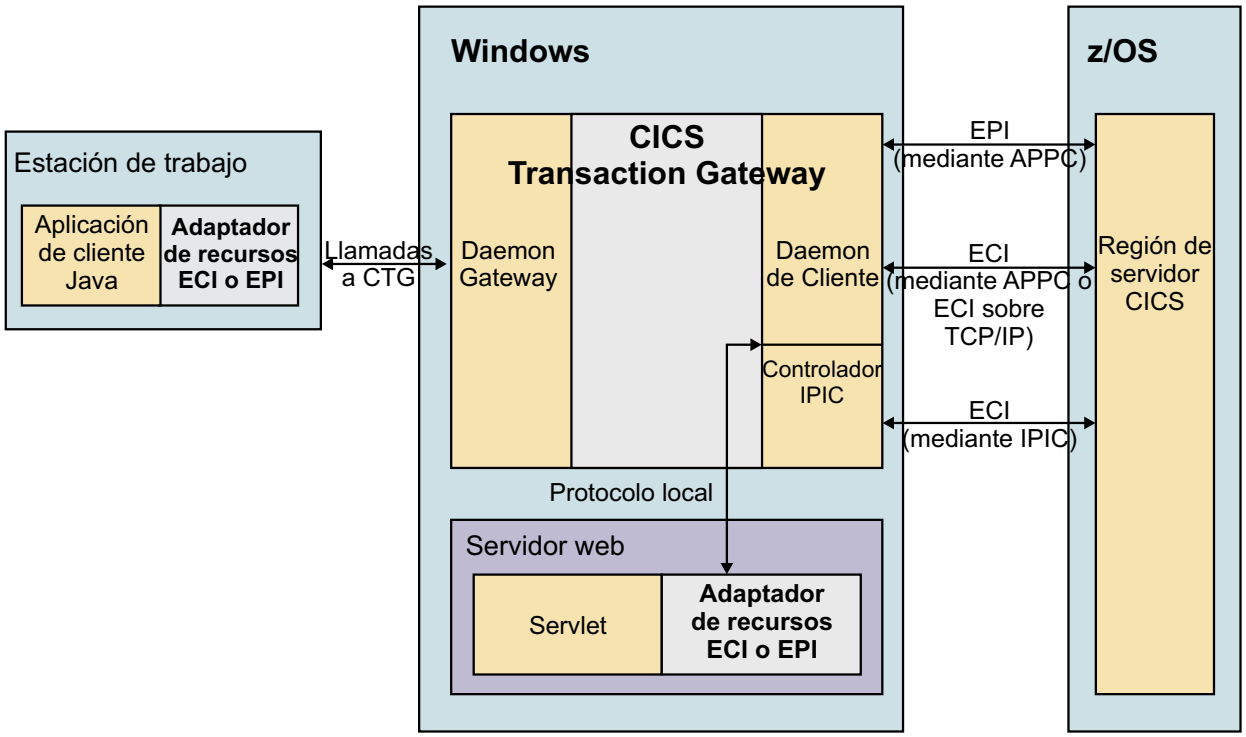


Figura 2. Los clientes Java se conectan con un programa de CICS desde fuera de CICS

Para utilizar programas de CICS de este modo, el desarrollador de Java necesita algún conocimiento sobre desarrollo de aplicaciones CICS.

CICS Transaction Gateway

- Guía de programación
- Escenarios
- Referencia de programación

IBM Redbooks

- Desarrollo de aplicaciones de conector para CICS
- Java Connectors for CICS Featuring the J2EE Connector Architecture

Servicios web de Java

CICS incluye la tecnología Axis2 para ejecutar servicios web de Java. Axis2 es un motor de servicios web de código abierto de la Apache Foundation y se proporciona con CICS para procesar mensajes SOAP en un entorno Java.

Axis2 es una implementación basada en Java de un motor SOAP de servicios web que soporta varias especificaciones de servicios web. También proporciona un modelo de programación que describe cómo crear aplicaciones Java que se ejecutan en Axis2. Axis2 se proporciona con CICS para procesar servicios web en un entorno Java. Aunque los tiempos de respuesta de Axis2 son algo inferiores a sus equivalentes que no son de Java, este tipo de carga de trabajo de Java es elegible para ejecutarse en un zAAP.

El servidor de JVM soporta la ejecución de Axis2 para procesar mensajes SOAP de entrada y salientes en una interconexión SOAP basada en Java sin modificar ninguno de los servicios web existentes. Sin embargo, también puede crear un servicio web a partir de una aplicación Java y ejecutarlo en el mismo servidor de JVM. Al desplegar la aplicación en el repositorio de Axis2 del servidor de JVM, tanto la aplicación Java como el proceso de SOAP son elegibles para su ejecución en un zAAP.

Tal vez desee utilizar servicios web de Java por una de las siguientes razones:

- Tiene experiencia con los servicios web de Axis2 en otras plataformas y desea crear servicios web en CICS.
- Desea utilizar API de Java estándar para crear enlaces de datos Java que se integren con Axis2.
- Tiene documentos WSDL complejos que son difíciles de manejar con los asistentes de servicios web de CICS.
- El usuario desea ejecutar el manejo de la aplicación de servicio web en un zAAP.

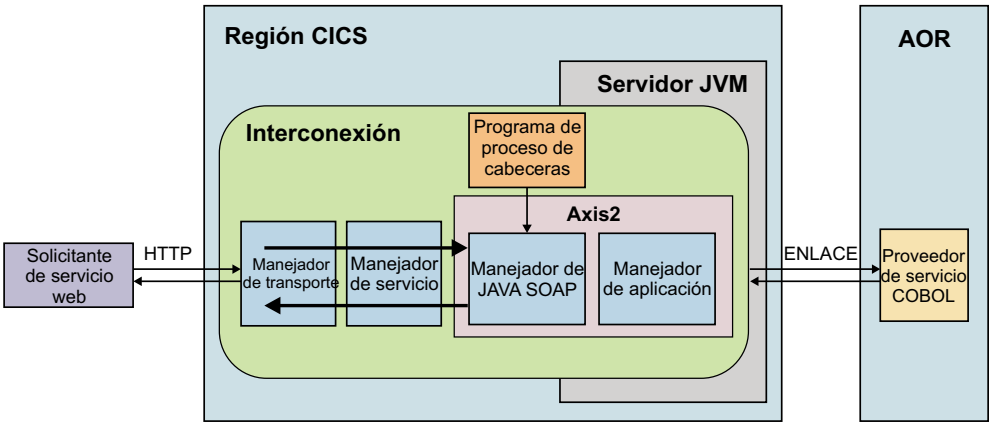
Los siguientes ejemplos describen cómo se puede utilizar Java con servicios web.

Proceso de mensajes SOAP en un servidor de JVM

La mayor parte del proceso de SOAP que se produce en la interconexión de servicios web lo realiza el manejador SOAP y el manejador de aplicación. Como opción, puede ejecutar este proceso de SOAP en un servidor de JVM y utilizar zAAP para ejecutar el trabajo. Puede continuar utilizando aplicaciones de servicios web que estén escritas en COBOL, C, C++ o PL/I.

Si cuenta con servicios web existentes, puede actualizar la configuración de sus interconexiones para utilizar un servidor de JVM. No tiene que realizar ningún otro cambio a los servicios web. Si la interconexión utiliza un programa de proceso de cabeceras SOAP, es mejor volver a escribir el programa en Java usando el modelo de programación Axis2. El programa de proceso de cabeceras puede compartir los objetos Java con Axis2 sin realizar más conversión de datos. Si tiene un programa de proceso de cabeceras en COBOL, por ejemplo, los datos se deben convertir de Java a COBOL y viceversa, lo que puede ralentizar el rendimiento del proceso SOAP.

El escenario que se muestra en el siguiente diagrama es un ejemplo de una aplicación COBOL que es un proveedor de servicios web. La solicitud se procesa en una interconexión que está configurada para el soporte de Java. El manejador SOAP y el manejador de aplicación son programas Java procesados por Axis2 y que se ejecutan en un servidor de JVM. El manejador de aplicación convierte los datos de XML a COBOL y enlaza con la aplicación.



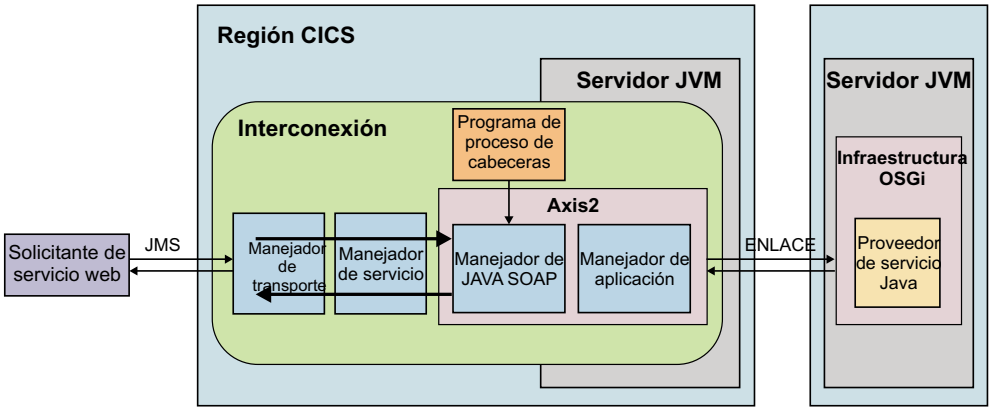
Si planifica su entorno, asegúrese de utilizar un conjunto de regiones dedicadas para sus servidores de JVM. En este ejemplo, la aplicación COBOL se ejecuta en una región propietaria de la aplicación (AOR) que es independiente de la región CICS en la que se ejecuta el servidor de JVM. Puede utilizar gestión de carga de trabajo para equilibrar las cargas de trabajo: por ejemplo, en el **EXEC CICS LINK** desde el manejador de aplicación, o en la solicitud de entrada desde el solicitante de servicio web.

Escritura de una aplicación Java que utilice salida desde el asistente de servicios web de CICS

Puede escribir una aplicación Java que interprete las estructuras de lenguaje y utilice los enlaces de datos generados por el asistente de servicios web de CICS. El asistente de servicios web puede producir estructuras de lenguaje desde WSDL o WSDL a partir de estructuras de lenguaje. El asistente también produce un enlace de servicio web que describe cómo convertir los datos entre XML y el lenguaje de destino durante el proceso SOAP.

Si se utiliza el asistente para generar una estructura de lenguaje, puede emplear JZOS o J2C para trabajar con las estructuras de lenguaje y generar clases Java. Estas herramientas proporcionan un modo de que los desarrolladores Java interaccionen con otras aplicaciones CICS. En este ejemplo, puede utilizar estas herramientas para escribir una aplicación Java que maneje un mensaje SOAP de entrada después de que CICS haya convertido los datos desde XML. Para obtener más información, consulte “Interacción con datos estructurados procedentes de Java” en la página 51.

El escenario que se muestra en el siguiente diagrama es un ejemplo de una aplicación Java que es un proveedor de servicios web. El proceso SOAP lo maneja Axis2 en un servidor de JVM. El manejador de aplicación enlaza con la aplicación Java, que se empaqueta y se despliega como uno o más paquetes OSGi y se ejecuta en un servidor de JVM.



La ventaja de este método es que, debido a que los enlaces de datos los generó el asistente de servicios web, el servicio web está representado en CICS por el recurso WEBSERVICE. Puede utilizar estadísticas, gestión de recursos y otros recursos en CICS para gestionar el servicio web. La desventaja es que el desarrollador Java debe trabajar con estructuras de lenguaje para un lenguaje de programación que puede no resultarle familiar.

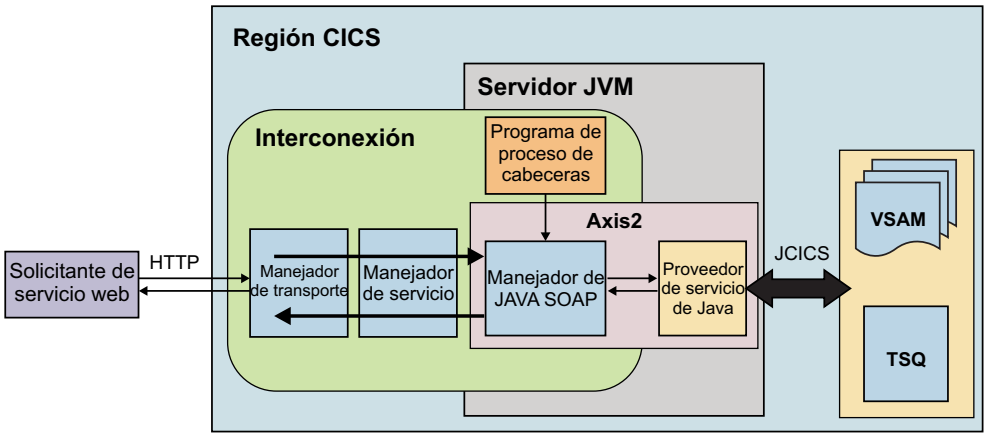
Cuando esté planificando su entorno para este tipo de aplicación, utilice un servidor de JVM distinto para ejecutar la aplicación:

- Puede gestionar y ajustar de forma más efectiva los servidores de JVM para las distintas cargas de trabajo.
- Puede utilizar gestión de carga de trabajo en las solicitudes de entrada y **EXEC CICS LINK** para equilibrar cargas de trabajo y escalar el entorno.
- Puede aprovecharse del soporte de OSGi en CICS para gestionar la aplicación Java.

Escritura de una aplicación Java que utilice enlaces de datos Java

Puede escribir una aplicación Java que genere y analice el XML para mensajes SOAP. La API de Java 6 ofrece bibliotecas Java estándar para trabajar con XML; por ejemplo, puede utilizar la arquitectura Java para enlaces XML (JAXB) para crear los enlaces de datos Java y las bibliotecas de la API de Java para servicios web XML (JAX-WS) para generar y analizar el XML. Si utiliza estas bibliotecas, la aplicación se puede ejecutar en Axis2 en el mismo servidor de JVM como el proceso de la interconexión SOAP.

El escenario que se muestra en el siguiente diagrama es un ejemplo de una aplicación Java que es un proveedor de servicios web y lo procesa el motor Axis2 SOAP en un servidor de JVM.



La aplicación Java utiliza enlaces de datos Java e interacciona con el manejador SOAP de Java, por lo que no hay manejador de aplicación. En este ejemplo, el solicitante de servicio web utiliza HTTP para conectar con la región CICS, pero también se puede emplear JMS. La aplicación Java utiliza JCICS para acceder a servicios de CICS; en este ejemplo, archivos VSAM y una cola de almacenamiento temporal.

La ventaja de este método es que el desarrollador de Java utiliza tecnologías que le resultan familiares para crear la aplicación. Asimismo, el desarrollador de Java puede trabajar con documentos WSDL complejos que el asistente de servicios web no puede procesar para producir un enlace. Sin embargo, este método tiene algunas limitaciones:

- No puede utilizar WS-Security para este tipo de aplicación, por lo que si desea utilizar seguridad, emplee SSL para asegurar la conexión.
- No se produce conmutador de contexto para el ID de usuario en el proceso de la interconexión. Para modificar el ID de usuario en la solicitud, utilice un recurso URIMAP.
- Como no utiliza el enlace de servicio web del asistente de servicios web, no hay ningún recurso WEBSERVICE.
- Si la aplicación es un solicitante de servicio web, el proceso de interconexión se elude, por lo que no se obtienen las calidades de servicio que están disponibles en la interconexión.

Si implementa gestión de carga de trabajo en las regiones CICS, debe planificar cómo dirigir este tipo de carga de trabajo. Puesto que la aplicación Java se ejecuta en el mismo servidor JVM que el proceso SOAP, CICS no proporciona ninguna oportunidad de direccionamiento. No obstante, puede implementar un enlace de programa distribuido en la aplicación JAX-WS para otro programa si se requiere el direccionamiento.

Aplicaciones Java compatibles con OSGi

CICS incluye la implementación Equinox de la infraestructura OSGi para ejecutar aplicaciones Java que cumplan con la especificación OSGi en un servidor de JVM.

La especificación de la Plataforma de servicios OSGi, como se describe en “La plataforma de servicios OSGi” en la página 2, proporciona una infraestructura para ejecutar y gestionar aplicaciones Java modulares y dinámicas. La configuración

predeterminada de un servidor de JVM incluye la implementación Equinox de una infraestructura OSGi. Las aplicaciones Java que se despliegan en la infraestructura OSGi de un servidor de JVM se benefician de las ventajas de utilizar OSGi y de las calidades de servicio que son inherentes a la ejecución de aplicaciones en CICS.

Tal vez desee utilizar aplicaciones Java por cualquiera de las siguientes razones:

- Desea crear cargas de trabajo de Java que se puedan ejecutar en un procesador zAAP para reducir el coste de las transacciones.
- Tiene experiencia con la escritura de aplicaciones Java que utilizan OSGi en otras plataformas y desea crear aplicaciones Java en CICS.
- Desea proporcionar aplicaciones Java como un conjunto de componentes modulares que se pueden reutilizar y actualizar de forma independiente, sin afectar a la disponibilidad de las aplicaciones y de la JVM en la que se ejecutan.

Para desarrollar, desplegar y gestionar de forma efectiva aplicaciones Java que sean compatibles con OSGi, debe utilizar el CICS Explorer SDK y el CICS Explorer:

- El CICS Explorer SDK amplía un entorno de desarrollo integrado (IDE) de Eclipse para proporcionar las herramientas y el soporte para ayudar a que los desarrolladores de Java creen y desplieguen aplicaciones Java en CICS. Utilice esta herramienta para convertir aplicaciones Java existentes en paquetes OSGi.
- El CICS Explorer es una herramienta de gestión de sistemas basada en Eclipse que proporciona a los administradores de sistema vistas para paquetes OSGi, servicios OSGi y los servidores de JVM en los que se ejecutan. Utilice esta herramienta para habilitar e inhabilitar aplicaciones Java, comprobar el estado de paquetes y servicios OSGi en la infraestructura y obtener algunas estadísticas preliminares sobre el rendimiento del servidor de JVM.

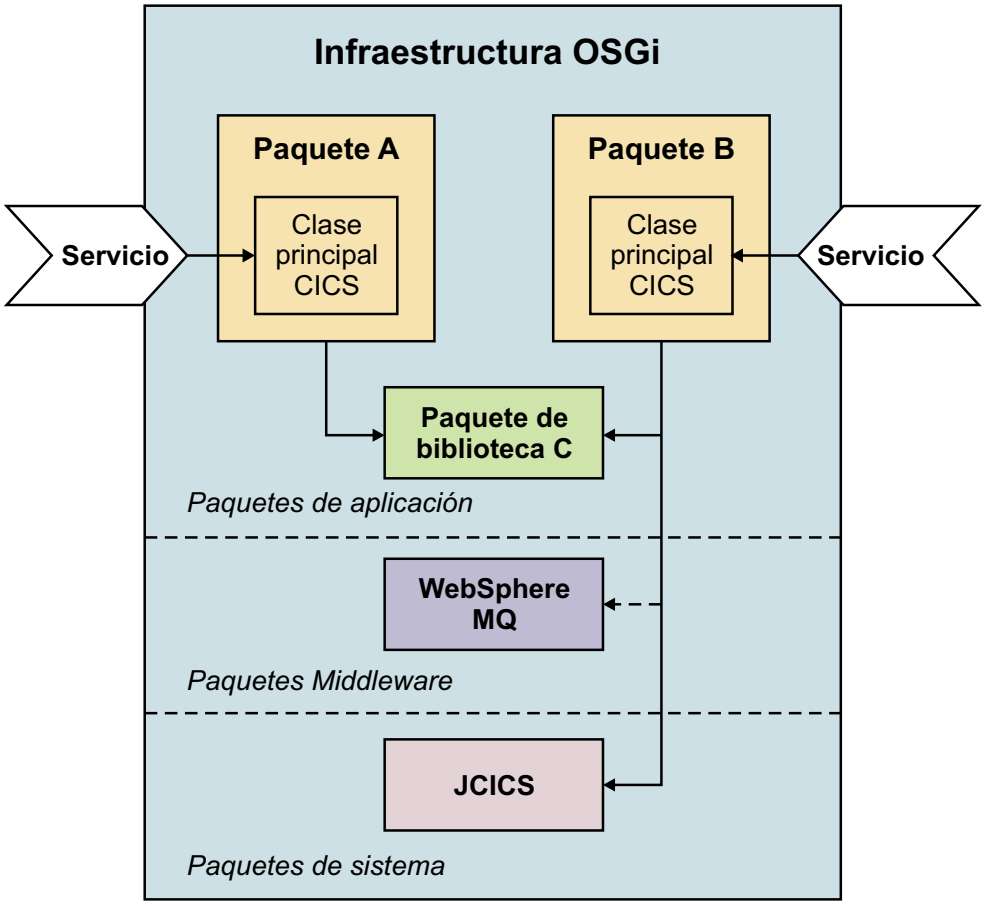
Cualquier desarrollador de Java o administrador de sistema que desee trabajar con OSGi tiene que acceder a estas herramientas gratuitas.

Los siguientes ejemplos describen cómo se pueden ejecutar aplicaciones Java que utilicen OSGi en CICS.

Ejecución de varias aplicaciones Java en el mismo servidor de JVM

El servidor de JVM puede gestionar varias solicitudes en la misma JVM de forma simultánea. Por lo tanto, puede llamar a la misma aplicación varias veces de forma simultánea o ejecutar varias aplicaciones en el mismo servidor de JVM.

Cuando haya decidido cómo dividir las aplicaciones entre servidores de JVM, podrá planificar cómo utilizar el modelo OSGi para subdividir las aplicaciones en un conjunto de paquetes OSGi. También debe decidir qué paquetes OSGi compatibles se necesitan en la infraestructura para ofrecer servicios a las aplicaciones. La infraestructura OSGi puede contener distintos tipos de paquete OSGi, como se muestra en el siguiente diagrama:



Paquetes de aplicación

Un paquete de aplicación es un paquete OSGi que contiene código de aplicación. Los paquetes OSGi pueden ser autónomos o tener dependencias de otros paquetes en la infraestructura. Estas dependencias las gestiona la infraestructura, de forma que un paquete OSGi que tenga una dependencia sin resolver no se puede ejecutar en la infraestructura. Para que la aplicación sea accesible fuera de la infraestructura en CICS, un paquete OSGi debe declarar una clase principal CICS como su servicio OSGi. Si un recurso PROGRAM apunta a la clase principal de CICS, otra aplicación fuera de la infraestructura OSGi puede acceder a la aplicación Java. Si cuenta con un paquete OSGi que contiene bibliotecas comunes para una o más aplicaciones, un desarrollador de Java puede decidir no declarar una clase principal de CICS. Este paquete OSGi está disponible solo para otros paquetes OSGi en la infraestructura.

La unidad de despliegue para una aplicación Java es un paquete CICS. Un paquete CICS puede contener cualquier número de paquetes OSGi y se puede desplegar en uno o más servidores de JVM. Puede añadir, actualizar y eliminar paquetes de aplicación independientemente de la gestión del servidor de JVM.

Paquetes de middleware

Un paquete de middleware es un paquete OSGi que contiene clases para implementar servicios del sistema, como conectar con WebSphere MQ. Otro ejemplo podría ser un paquete OSGi que contenga código nativo y deba cargarse solo una vez en la infraestructura OSGi. Un paquete de

middleware se gestiona con el ciclo de vida del servidor de JVM, en lugar de con las aplicaciones que utilizan sus clases. Los paquetes de middleware se especifican en el perfil de JVM del servidor de JVM y los carga CICS cuando dicho servidor de JVM se lanza.

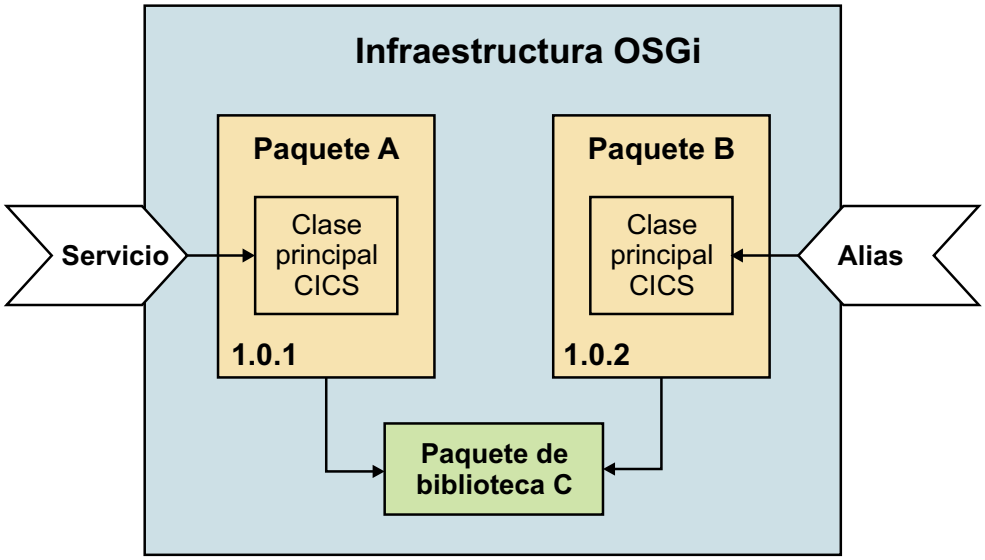
Paquetes del sistema
Un paquete del sistema es un paquete OSGi que gestiona la interacción entre CICS y la infraestructura OSGi para proporcionar servicios clave para las aplicaciones. El ejemplo primario son los paquetes OSGi de JCICS, que brindan acceso a servicios y recursos de CICS.

Para simplificar la gestión de las aplicaciones Java, siga estas prácticas recomendadas:

- Despliegue paquetes OSGi de acoplamiento hermético que consten de una aplicación en el mismo paquete CICS. Los paquetes de acoplamiento hermético exportan clases directamente de unos a otros sin utilizar servicios OSGi. Despliegue estos paquetes OSGi juntos en un paquete CICS para actualizarlos y gestionarlos a la vez.
- Evite crear dependencias entre aplicaciones. En vez de eso, cree una biblioteca común en un paquete OSGi distinto y gestiónela en su propio paquete CICS. Es posible actualizar la biblioteca desde las aplicaciones.
- Siga las prácticas recomendadas de OSGi utilizando versiones al crear dependencias entre paquetes. La utilización de un rango de versiones significa que una aplicación puede tolerar actualizaciones compatibles para los paquetes de los que depende.
- Configure un convenio de denominación para los servidores de JVM y acuerde el convenio entre los programadores del sistema y los desarrolladores de Java.

Ejecución de varias versiones de una misma aplicación Java en un servidor de JVM

La infraestructura OSGi soporta la ejecución de varias versiones de un paquete OSGi en una infraestructura, de manera que pueda realizar actualizaciones graduales en la aplicación sin interrumpir su disponibilidad. Sin embargo, no puede contar con varias versiones del mismo servicio OSGi en la infraestructura. Si distintas versiones del paquete OSGi tienen la misma clase principal CICS, puede utilizar un alias para sustituir el servicio duplicado. El alias se especifica con la declaración de la clase principal CICS y se registra en la infraestructura OSGi como el servicio OSGi para la versión actualizada del paquete. Especifique el alias en otro recurso PROGRAM para hacer que la aplicación esté disponible.


Infraestructura OSGi
Paquete A
Clase principal CICS
1.0.1
Paquete B
Clase principal CICS
1.0.2
Servicio
Alias
Paquete de biblioteca C

Capítulo 3. Desarrollo de aplicaciones Java para CICS

Puede escribir programas de aplicación Java que utilicen servicios de CICS y que se ejecuten bajo el control de CICS. Mediante el CICS Explorer SDK, puede desarrollar aplicaciones que utilicen la biblioteca de clases de JCICS para acceder a recursos de CICS e interaccionar con programas que estén escritos en otros lenguajes. También puede conectar con los programas Java utilizando varios protocolos y tecnologías, como servicios web o CICS Transaction Gateway.

CICS proporciona herramientas y el entorno de ejecución para el soporte de aplicaciones Java. El CICS Explorer SDK es una herramienta basada en Eclipse que brinda soporte para el desarrollo y el despliegue de aplicaciones Java en CICS. Contiene la biblioteca de clases JCICS para desarrollar aplicaciones que accedan a recursos y servicios de CICS; por ejemplo, puede acceder a archivos VSAM, colas de datos transitorias y almacenamiento temporal. También puede utilizar JCICS para enlazar con aplicaciones de CICS escritas en otros lenguajes, como COBOL o C.

El CICS Explorer SDK proporciona otras funciones, como el empaquetado de aplicaciones para que cumplan con la especificación OSGi y brindar un entorno de destino para asegurar que se utilicen únicamente las clases que están soportadas en una versión específica de CICS. También se incluyen aplicaciones de ejemplo de JCICS para ayudarle a empezar si acaba de iniciarse en el desarrollo de aplicaciones Java para CICS.

Qué necesita saber acerca de CICS

CICS es un subsistema de proceso de transacción que brinda servicios para que un usuario ejecute aplicaciones por solicitud, a la vez que otros muchos usuarios envían solicitudes para ejecutar las mismas aplicaciones utilizando los mismos archivos y programas. CICS gestiona la compartición de recursos, la integridad de los datos y la priorización de la ejecución, a la vez que mantiene tiempos de respuesta rápidos.

Una aplicación de CICS es un conjunto de programas relacionados que juntos realizan una operaciones de negocio, como el proceso de un pedido de producto o la preparación de la nómina de una compañía. Las aplicaciones de CICS se ejecutan bajo control de CICS utilizando servicios e interfaces de CICS para acceder a programas y archivos.

Las aplicaciones de CICS se ejecutan enviando una solicitud de *transacción*. El término transacción tiene un significado especial en CICS; consulte “Transacciones de CICS” en la página 30 para obtener una explicación de las diferencias entre el uso de CICS y el uso más común en la industria. La ejecución de la transacción consiste en ejecutar uno o más programas de aplicación que implementan la función necesaria.

Para desarrollar aplicaciones Java para CICS debe comprender la relación entre programas, transacciones y tareas de CICS. Estos términos se utilizan en la documentación de CICS y aparecen en muchos mandatos de programación. También tiene que comprender la manera en la que CICS maneja las aplicaciones Java en el entorno de ejecución.

Transacciones de CICS

Una transacción es un fragmento de proceso lanzado por una única solicitud.

La solicitud generalmente la realiza un usuario en un terminal. Sin embargo, se podría realizar desde una página web, desde el programa de una estación de trabajo remota o desde una aplicación en otra región CICS; o podría desencadenarse automáticamente a una hora predefinida. Los documentos Descripción general: CICS y HTTP en la Guía de Internet y Guía de interfaces externas en la Guía de interfaces externas describen distintas maneras de ejecutar transacciones de CICS.

Una única transacción consta de uno o más *programas de aplicación* que, cuando se ejecutan, llevan a cabo el proceso necesario.

Sin embargo, el término *transacción* se utiliza en CICS con el significado de un único suceso y también todas las demás transacciones del mismo tipo. Cada tipo de transacción se describe para CICS con una definición de recurso TRANSACTION. Esta definición brinda un nombre para el tipo de transacción (el identificador de transacción o TRANSID) e indica a CICS varias cosas sobre el trabajo a realizar, como qué programas invocar primero y qué tipo de autenticación es necesaria durante la ejecución de la transacción.

Se ejecuta una transacción sometiendo su TRANSID a CICS. CICS utiliza la información registrada en la definición TRANSACTION para establecer el entorno de ejecución correcto y lanza el primer programa.

El término *transacción* actualmente se utiliza ampliamente en la industria de las tecnologías de la información para describir una *unidad de recuperación* o lo que CICS llama una *unidad de recuperación*. Esto normalmente es una operación lógica completa que es recuperable, se puede confirmar o retrotraer como una totalidad, como resultado de un mandato programado o de una anomalía del sistema. En muchos casos, el ámbito de una transacción de CICS es también una única unidad de trabajo, pero debería conocer la diferencia de significado al leer la documentación de CICS.

Tareas de CICS

Una tarea es una instancia única de la ejecución de una transacción.

Esta palabra, *tarea*, tiene un significado específico en CICS. Cuando CICS recibe una solicitud para ejecutar una transacción, se lanza una nueva tarea que esté asociada con esta *única instancia* de la ejecución del tipo de transacción. Es decir, una tarea de CICS es una única ejecución de una transacción, con su propio conjunto de datos privado, generalmente en nombre de un usuario específico. También se puede considerar una tarea como una *hebra*. Las tareas las *asigna* CICS en función de su prioridad y de su grado de preparación. Cuando la transacción se completa, la tarea se termina.

Programas de aplicación de CICS

En los programas Java puede utilizar la biblioteca de clases Java para CICS (JCICS) para acceder a servicios de CICS y enlazar con los programas de aplicación que se escriben en otros lenguajes.

Los programas de aplicación de CICS pueden escribirse en COBOL, C, C++ , Java, PL/I o lenguajes ensambladores. La mayor parte de la lógica del proceso se

expresa en sentencias de lenguaje estándar, pero para solicitar servicios de CICS, las aplicaciones utilizan las interfaces de programación de aplicaciones provistas. COBOL, C, C++, PL/I o los programas ensambladores pueden utilizar la interfaz de programación de aplicaciones **EXEC CICS** o la biblioteca de clases de C++. Los programas de Java utilizan la biblioteca de clases de JCICS JCICS se describe en "La biblioteca de clases de Java para CICS (JCICS)" en la página 53.

Servicios de CICS

Los programas de Java pueden acceder a los siguientes servicios de CICS a través de la interfaz de programación JCICS: Gestión de datos, comunicaciones, unidad ed trabajo, programa y servicios de diagnóstico.

Los administradores de servicios de CICS generalmente tiene el control de palabras en su título; por ejemplo, "control de terminales" y "control de programa". Estos términos se utilizan ampliamente en la información de CICS.

Servicios de gestión de datos

CICS proporciona los siguientes servicios de gestión de datos:

- Compartición de nivel de registro, al acceder a conjuntos de datos de Virtual Storage Access Method (VSAM). CICS registra la actividad para dar soporte a la restitución de datos (para una anomalía de transacción o del sistema) y a la recuperación hacia delante (para una anomalía de soporte). El control de archivos de CICS gestiona los datos de VSAM.

  CICS también implementa dos estructuras de archivos propias y brinda mandatos para manipularlos:

  Almacenamiento temporal
  : El almacenamiento temporal (TS) es un medio de hacer que los datos estén fácilmente disponibles para varias transacciones. Los datos se mantienen en colas, que se crean según las necesiten los programas. El acceso a las colas puede ser secuencial o por número de elemento.

    Las colas de almacenamiento temporal pueden residir en la memoria principal o grabarse en un dispositivo de almacenamiento.

    Una cola de almacenamiento temporal se puede considerar una anotación con nombre.

  Datos transitorios
  : También hay datos transitorios (TD) disponibles para varias transacciones y se mantienen en colas. Sin embargo, a diferencia de las colas de TS, las colas de TD deben predefinirse y solo se pueden leer de forma secuencial. Cada elemento se elimina de la cola cuando se lee.

    Las colas de datos transitorias siempre se graban en un conjunto de datos. Puede definir una cola de datos transitoria de forma que la grabación de un número específico de elementos en ella pueda actuar como desencadenante para iniciar una transacción específica. Por ejemplo, la transacción desencadenada puede procesar la cola.
- El acceso a datos en otras bases de datos (incluida DB2), a través de interfaces con productos de base de datos.

Servicios de comunicaciones

CICS proporciona mandatos que brindan acceso a una amplia gama de terminales (pantallas, impresoras y estaciones de trabajo) utilizando los protocolos SNA y TCP/IP. El control de terminales de CICS proporciona la gestión de redes SNA y TCP/IP.

Puede escribir programas que utilicen mandatos de Comunicación Avanzada Programa a Programa (APPC) para lanzar otros programas en sistemas remotos y comunicarse con ellos, mediante protocolos SNA. La APPC de CICS implementa el modelo de aplicación distribuida de igual a igual.

CICS también proporciona un intermediario para solicitudes de objetos (ORB) para implementar los protocolos Inter-ORB de Internet de entrada y salientes definidos por la Common Object Request Broker Architecture (CORBA). El ORB soporta solicitudes para ejecutar objetos sin estado Java y enterprise beans.

Se proporcionan los siguientes servicios de comunicaciones reservados de CICS:

Envío de funciones
: Las solicitudes de programas para acceder a recursos (archivos, colas y programas) que se definen como existentes en regiones CICS remotas son dirigidas automáticamente por CICS a la región propietaria.

Enlace de programa distribuido (DPL)
: Las solicitudes de enlace de programa para un programa definido como existente en una región CICS remota se dirigen automáticamente a la región propietaria. CICS proporciona mandatos para mantener la integridad de la aplicación distribuida.

Proceso asíncrono
: CICS proporciona mandatos para permitir que un programa lance otra transacción en la misma región CICS o en una remota y, como opción, pasarle datos. La nueva transacción se planifica de forma independiente en una nueva tarea. Esta función es similar a la operación de *bifurcación* proporcionada por otros productos de software.

Direccionamiento de transacción
: Las solicitudes para ejecutar transacciones definidas como existentes en regiones CICS remotas se dirigen automáticamente a la región propietaria. Las respuestas para el usuario se dirigen de vuelta a la región que recibiera la solicitud.

Servicios de unidad de trabajo

Cuando CICS crea una nueva tarea para ejecutar una transacción, se inicia una nueva unidad de trabajo (UOW) automáticamente. (Por eso CICS no proporciona un mandato BEGIN, ya que no es necesario ninguno.) Las transacciones de CICS siempre se ejecutan en transacción.

CICS proporciona un mandato SYNCPOINT para confirmar o retrotraer el trabajo recuperable realizado. Cuando se completa el punto de sincronización, CICS inicia automáticamente otra unidad de trabajo. Si se termina el programa sin emitir un mandato SYNCPOINT, CICS toma un punto de sincronización implícito e intenta confirmar la transacción.

El ámbito de la confirmación incluye todos los recursos de CICS que se han definido como recuperables y cualquier otro Gestor de recursos que haya registrado un interés mediante las interfaces provistas por CICS.

Si escribe enterprise beans mediante servicios de transacción proporcionados por los mandatos definidos por el Java Transaction Service (JTS), CICS correlaciona estos mandatos (incluido BEGIN) con sus servicios de unidad de trabajo.

Servicios de programa

CICS proporciona mandatos que permiten a un programa enlazar con otro programa o transferirle el control, y luego volver.

Servicios de diagnóstico

CICS proporciona mandatos que puede utilizar para rastrear programas y producir volcados.

Entorno de tiempo de ejecución Java en CICS

CICS proporciona dos entornos de ejecución para ejecutar aplicaciones Java. Las aplicaciones de enhebramiento seguro pueden utilizar un servidor de JVM. Las aplicaciones que no son de enhebramiento seguro tienen que utilizar JVM en agrupación.

Servidores de JVM

El servidor de JVM es un entorno de ejecución que puede ejecutar tareas en una única JVM. Este entorno es el preferido para ejecutar aplicaciones Java, porque reduce el almacenamiento virtual necesario para cada tarea Java y permite a CICS ejecutar muchas tareas de forma simultánea.

Las tareas de CICS se ejecutan en paralelo como hebras en el mismo proceso de servidor de JVM. Todas las tareas de CICS, que pueden ejecutar varias aplicaciones a la vez, no solo comparten la JVM, sino que también comparten todos los datos estáticos y las clases estáticas. Así que para utilizar un servidor de JVM en CICS, una aplicación Java debe ser de enhebramiento seguro. Cada hebra se ejecuta en un TCB T8 y puede acceder a servicios de CICS mediante la API de JCICS.

Puede escribir código de aplicación para lanzar una nueva hebra o llamar a una biblioteca que lance una hebra. Sin embargo, estas hebras no pueden acceder a servicios de CICS. Cualquier intento por acceder a servicios de CICS desde una hebra generada por aplicación producirá un error Java bm.exception. Si desea crear hebras en la aplicación, asegúrese de que estas no se ejecutan más allá del tiempo de vida de la tarea de CICS que ejecuta la aplicación. Cuando el programador del sistema inhabilita el servidor de JVM, CICS espera a que todas las hebras actuales que se ejecutan en TCB T8 finalicen en la JVM. Sin embargo, las hebras creadas por la propia aplicación se terminan.

Como las clases y los datos estáticos los comparten todas las hebras que se ejecutan en el servidor de JVM, puede crear clases de activadores de paquetes OSGi para inicializar los datos estáticos y dejarlos en el estado correcto cuando se apague la JVM. Un servidor de JVM se ejecuta hasta que el programador del sistema lo inhabilita: por ejemplo, para añadir una aplicación o arreglar un problema. Al proporcionar clases de activadores de paquetes OSGi, puede asegurarse de que el estado se define correctamente para sus aplicaciones. CICS

tiene un tiempo de espera que especifica cuánto se debe esperar a que estas clases se completen antes de continuar con el inicio o la parada del servidor de JVM. No se puede utilizar JCICS en las clases de inicio y terminación.

No utilice el método System.exit() en sus aplicaciones. Este método hace que el servidor de JVM y CICS se apaguen, lo cual afecta al estado y disponibilidad de las aplicaciones.

JVM en agrupación

Una JVM en agrupación puede gestionar únicamente una sola solicitud para una sola aplicación Java en cada momento, por lo que son necesarias muchas más JVM en una región CICS. Una JVM en agrupación está aislada, por lo que una aplicación Java no tiene que ser de enhebramiento seguro. Sin embargo, las JVM en agrupación generalmente se reutilizan muchas veces, normalmente por parte de distintas aplicaciones, por lo que es importante mantener el aislamiento de transacción y el estado de los datos.

La hebra principal en en la que se lanza una JVM se denomina la hebra de proceso inicial (IPT). CICS se asegura de que el método principal estático público de cualquier programa Java se ejecute en la IPT de una JVM en agrupación. Si desea crear hebras en la aplicación, estas no deben intentar acceder a servicios de CICS y no se deben ejecutar más allá del tiempo de vida de la tarea de CICS que inicia las hebras. Si las hebras de usuario se siguen ejecutando una vez que la IPT ha devuelto el control a CICS, estas hebras pueden dañar el aislamiento para la JVM cuando esta sea reutilizada por otra aplicación, así como provocar problemas cuando CICS intente detener la JVM.

Instalación del CICS Explorer SDK

El CICS Explorer SDK está disponible de forma gratuita para su descarga del sitio web de IBM para instalarlo en un entorno de desarrollo integrado (IDE) de Eclipse.

Antes de empezar

Debe tener el software necesario instalado en su estación de trabajo. La lista de sistemas operativos y de software se describen en el sitio web CICS Explorer.

Acerca de esta tarea

El CICS Explorer SDK es una infraestructura basada en Eclipse para desarrollar ampliaciones para el CICS Explorer. También brinda soporte para el desarrollo de aplicaciones Java para ejecutar en cualquier versión soportada de CICS. Brinda soporte para JCICS y para el empaquetado de aplicaciones para que cumplan con las especificaciones OSGi.

Procedimiento

1. Vaya al sitio web de CICS Explorer.
2. Seleccione el enlace **Download site** (Sitio de descargas) y escriba su identificador y su contraseña de IBM.
3. Seleccione CICS Explorer de la lista y pulse en **Continue** (Continuar).
4. Lea y acepte la licencia.
5. Seleccione el CICS Explorer SDK de la lista para descargar el archivo comprimido a un directorio de la estación de trabajo.

6. Abra el IDE de Eclipse y pulse en **Help** > **Install new software** (Ayuda > Instalar nuevo software).
7. Pulse en **Add** (Añadir). En el cuadro de diálogo Add site (Añadir sitio), pulse en **Archive** (Archivo).
8. Examine el archivo descargado y pulse en **Open** (Abrir).
9. Seleccione el recuadro de selección junto al kit de desarrollo de software de IBM CICS Explorer y pulse en **Next** (Siguiente).
10. Acepte la licencia y pulse **Finalizar** para instalar CICS Explorer SDK.

Resultados

El CICS Explorer SDK se instala en el IDE de Eclipse. Es posible que deba aceptar un aviso de seguridad y reiniciar el entorno de desarrollo integrado para recoger el software nuevo.

Qué hacer a continuación

Puede trabajar con los ejemplos de CICS que proporciona el CICS Explorer SDK para familiarizarse con el soporte de Java. Para obtener más información, consulte el apartado "Cómo empezar con los de ejemplo de JCICS".

Cómo empezar con los de ejemplo de JCICS

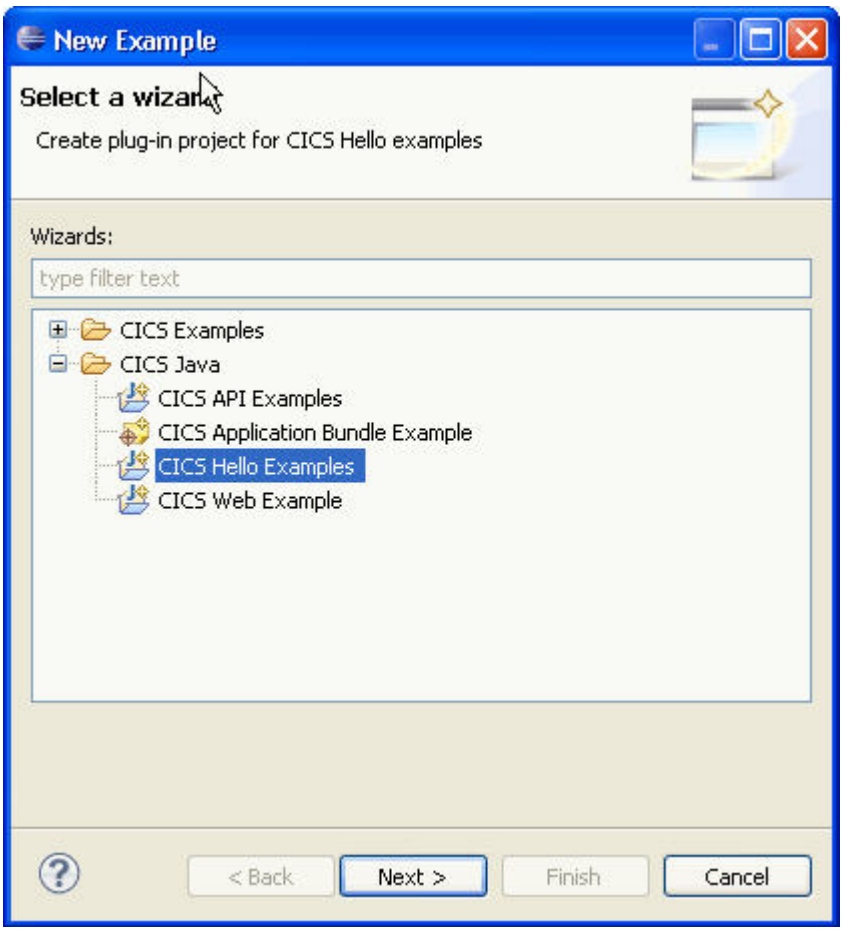
El CICS Explorer SDK contiene ejemplos de JCICS para ayudarle a empezar a desarrollar aplicaciones Java para CICS.

Acerca de esta tarea

Los ejemplos de JCICS se empaquetan como un conjunto de paquetes OSGi que se pueden importar a un proyecto de plug-in de Eclipse para visualizar el código fuente de Java. También puede utilizar la ayuda contextual para buscar las explicaciones de Javadoc relativas a los métodos que se utilizan en el código.

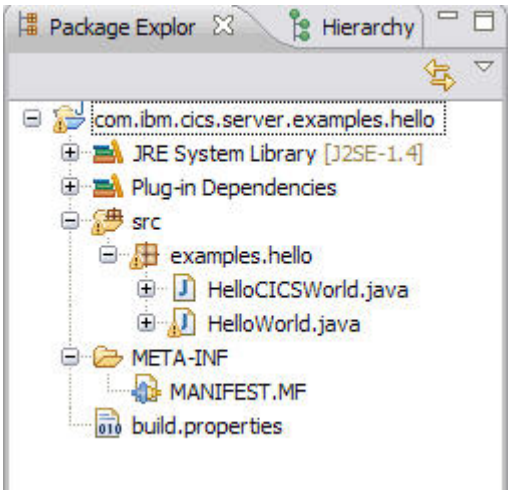
Procedimiento

1. En el IDE de Eclipse, abra la perspectiva Java.
2. Para crear un proyecto de plug-in de ejemplo, abra el asistente Nuevo ejemplo mediante una de las siguientes opciones:
   - En la barra de menús de Eclipse, pulse en **File** > **New** > **Example** (Archivo > Nuevo > Ejemplo).
   - Pulse en la flecha hacia abajo del icono **New Wizard** (Asistente nuevo) y haga clic en **Example** (Ejemplo).
   - En la vista del explorador de proyectos, pulse con el botón derecho del ratón y pulse en **New** > **Example** (Nuevo > Ejemplo).
3. En la carpeta **CICS Java**, seleccione**CICS Hello Examples** (Ejemplos de CICS Hello) y pulse en **Siguiente**.



- Los ejemplos de la API de CICS enseñan cómo utilizar colas de datos transitorias, colas de almacenamiento temporal y canales y COMMAREA en programas Java.
- El ejemplo de paquete de aplicación de CICS enseña cómo crear un paquete de CICS para desplegar en CICS.
- Los ejemplos de CICS hello muestran dos formas de realizar una prueba simple de Hello World en CICS.
- El ejemplo web de CICS enseña cómo utilizar clases para interaccionar con un navegador web.

4. En el campo **Project name** (Nombre del proyecto), escriba un nombre para el nuevo proyecto. De forma predeterminada, Eclipse crea un nombre que es la ubicación de la carpeta de los ejemplos en el espacio de trabajo seguido por el nombre del ejemplo. Por ejemplo, el nombre de proyecto predeterminado para el ejemplo Hello World es `com.ibm.cics.server.examples.hello`.
5. Pulse en **Finish** (Finalizar). Eclipse crea el proyecto de plug-in que contiene el ejemplo Hello World de JCICS como un paquete OSGi.
6. Expanda el proyecto en la vista del explorador de paquetes.



- La carpeta **Plug-in Dependencies** (Dependencias de plug-in) contiene las dependencias para el paquete OSGi. En este ejemplo, el paquete tiene una dependencia en el paquete OSGi que contiene JCICS. Esta información también se captura en el manifiesto del proyecto.
- La carpeta **src** contiene el origen de Java para los ejemplos. Puede examinar los archivos de origen para ver las clases JCICS que se utilizan y utilizar la ayuda contextual para localizar una clase concreta. También puede abrir la vista Javadoc para ver los detalles de la API del contenido seleccionado; por ejemplo, un método o una clase.
- La carpeta **META-INF** contiene el manifiesto para el proyecto. El manifiesto contiene las cabeceras OSGi para describir el paquete OSGi.

7. Cree proyectos plug-in para la API de CICS y ejemplos web de CICS mediante el asistente Nuevo ejemplo. Puede ver el origen de Java para comprender cómo se utilizan las clases JCICS para trabajar con programas o aplicaciones web.

Resultados

Ha creado tres proyectos plug-in en Eclipse para los ejemplos de JCICS. Estos proyectos contienen información de empaquetado de paquetes OSGi, como las dependencias de plug-in y los entornos Java de destino.

Qué hacer a continuación

Para ejecutar aplicaciones Java en CICS, debe desplegar la aplicación Java en un proyecto de paquete de CICS para zFS. Puede probar el proceso de despliegue utilizando los ejemplos de JCICS, como se describe en "Despliegue de los ejemplos de JCICS".

Despliegue de los ejemplos de JCICS

Puede utilizar el paquete de CICS de ejemplo del CICS Explorer SDK para desplegar los ejemplos de JCICS en una región CICS.

Antes de empezar

Debe haber creado los proyectos de ejemplo de JCICS, como se describe en "Cómo empezar con los de ejemplo de JCICS" en la página 35.

Acerca de esta tarea

CICS carga y ejecuta aplicaciones Java desde zFS, por lo que debe desplegar las aplicaciones compiladas en un directorio adecuado de zFS. Puede crear un directorio adecuado en zFS utilizando la perspectiva z/OS en CICS Explorer. CICS debe poseer acceso de lectura y ejecución para este directorio.

El CICS Explorer SDK proporciona soporte para desplegar aplicaciones Java en un proyecto de paquete CICS en zFS. Un proyecto de paquete CICS agrupa un conjunto de paquetes OSGi entre sí que se despliegan y se gestionan de forma lógica como una única unidad. Puede utilizar el proyecto de paquete CICS de ejemplo para desplegar los ejemplos de JCICS en una región CICS.

Procedimiento

1. En el IDE de Eclipse, abra la perspectiva Java.
2. Abra el asistente Nuevo ejemplo mediante una de las siguientes opciones:
   - En la barra de menús de Eclipse, pulse en **File** > **New** > **Example** (Archivo > Nuevo > Ejemplo).
   - Pulse en la flecha hacia abajo del icono **New Wizard** (Asistente nuevo) y haga clic en **Example** (Ejemplo).
   - En la vista del explorador de proyectos, pulse con el botón derecho del ratón y pulse en **New** > **Example** (Nuevo > Ejemplo).
3. En la carpeta **CICS Java**, seleccione **CICS Application Bundle Example** (Ejemplo de paquete de aplicaciones de CICS) y pulse en **Next** (Siguiente).
4. En el campo **Project name** (Nombre del proyecto), escriba un nombre para el nuevo proyecto. De forma predeterminada, Eclipse crea un nombre que es la ubicación de la carpeta de los ejemplos en el espacio de trabajo seguido por el nombre del ejemplo. Por ejemplo, el nombre de proyecto predeterminado para el paquete CICS es `com.ibm.cics.server.examples`.
5. Pulse en **Finish** (Finalizar). Eclipse crea el proyecto de paquete CICS que contiene un manifiesto y tres recursos. Estos recursos hacen referencia a tres paquetes OSGi.
6. Abra el archivo `web.osgibundle` para comprobar su contenido. Este archivo está en formato XML y contiene el nombre simbólico y la versión del paquete OSGi. También contiene el nombre de un servidor de JVM de ejemplo. El servidor de JVM es el entorno de ejecución para aplicaciones Java en CICS.Cuando cree sus propias aplicaciones, debe proporcionar el nombre del servidor JVM de destino en este archivo.
7. Despliegue el paquete de CICS en zFS:
   a. Pulse con el botón derecho del ratón en el proyecto de paquete CICS y seleccione **Export to z/OS UNIX File System** (Exportar al sistema de archivos z/OS UNIX).
   b. Escriba sus credenciales de FTP si es necesario. Tal vez necesite crear una conexión con una máquina host de destino si no ha configurado una conexión anteriormente.
   c. Examine un directorio en el que desee desplegar el paquete CICS y pulse en **Finish** (Finalizar).

   El paquete CICS se despliega en el directorio especificado.
8. Abra la perspectiva del CICS SM. En la vista de CICSplex Explorer, seleccione la región CICS en la que desea ejecutar los programas de ejemplo de JCICS.
9. Instale el recurso de JVMSERVER, DFH$JVMS, que está en el grupo de ejemplo DFH$OSGI. El recurso crea un servidor de JVM de ejemplo en la

región CICS que contiene una infraestructura OSGi. El nombre de este recurso coincide con el nombre del servidor de JVM que se especificara en el manifiesto del paquete CICS. Puede comprobar el estado del servidor de JVM pulsando en **Operations** > **Java** > **JVM Servers** (Operaciones > Java > Servidores de JVM).

10. Abra la vista Definiciones de paquete pulsanso en **Definiciones** > **Definiciones de paquete**. Esta vista incluye todas las definiciones de paquete de la región CICS.
11. En la vista Definiciones de grupo de recursos, seleccione el grupo DFH$OSGI proporcionado. Si esta vista no se abre, seleccione **Window** > **Show view** (Ventana > Mostrar vista) para abrirla en la perspectiva de Eclipse. La vista Bundle Definitions (Definiciones de paquete) se filtra para mostrar la definición de recurso DFH$OSGB.
12. Copie la definición de recurso en un nuevo grupo para editar los atributos:
    a. Pulse con el botón derecho del ratón en DFH$OSGB y seleccione **Copiar**.
    b. Pulse con el botón derecho del ratón en cualquier lugar de la vista Resource Group Definitions (Definiciones de grupo de recursos) y seleccione **Pegar**.
    c. Escriba un nombre de grupo nuevo y haga clic en **Aceptar**.
13. Edite la definición de recurso BUNDLE del nuevo grupo para cambiar el directorio del paquete para que coincida con la ubicación del paquete CICS desplegado.
14. Pulse con el botón derecho del ratón en la definición para instalar el recurso BUNDLE. Puede comprobar el recurso BUNDLE instalado en el estado ENABLED (habilitado) pulsando en **Operations** > **Bundles** (Operaciones > Paquetes). También puede comprobar la lista de paquetes OSGi pulsando en **Operations** > **Java** > **OSGi Bundles** (Operaciones > Java > Paquetes OSGi).
15. Para ejecutar los ejemplos en un servidor de JVM, instale el grupo DFH$OSGI en la región CICS. Este grupo contiene las definiciones de recurso para todos los ejemplos. Los recursos BUNDLE y JVMSERVER de ejemplo no están instalados porque ya ha creado recursos del mismo nombre. Cuando se instala el grupo, CICS crea los recursos que son necesarios para ejecutar los ejemplos.

Resultados

Ha desplegado correctamente el paquete CICS de ejemplo en zFS y ha creado los recursos de CICS que son necesarios para ejecutar los ejemplos de JCICS.

Qué hacer a continuación

Puede ejecutar los ejemplos de JCICS, como se describe en "Ejecución de los ejemplos de JCICS".

Ejecución de los ejemplos de JCICS

CICS proporciona varios ejemplos de JCICS que se pueden ejecutar en CICS. Puede ejecutar los ejemplos en un servidor de JVM, el entorno preferido para la ejecución de aplicaciones Java, o puede ejecutarlos en una JVM en agrupación.

Antes de empezar

Los ejemplos de JCICS se deben desplegar en un directorio de zFS para el que la región CICS tenga acceso de lectura y ejecución.

Procedimiento

1. Asegúrese de que la región CICS está configurada correctamente.
   - Si desea ejecutar los ejemplos en un servidor de JVM, el grupo DFH$OSGI debe estar instalado en la región CICS. En concreto, el recurso DFH$JVMS debe estar habilitado en la región CICS. Este recurso es el servidor de JVM de ejemplo que se proporciona y utiliza el perfil DFHOSGI predeterminado. El recurso BUNDLE que contiene los paquetes OSGi para los ejemplos de JCICS también debe estar habilitado.
   - Si desea ejecutar los ejemplos en una JVM en agrupación, el grupo DFH$JVM debe estar instalado en la región CICS. Edite el perfil de JVM predeterminado DFHJVMPR para añadir la vía de acceso de clases `/usshome/samples/dfjcics` a la opción `CLASSPATH_SUFFIX`, donde *usshome* representa el valor del parámetro de inicialización del sistema **USSHOME**.
2. Siga el procedimiento adecuado para ejecutar cada ejemplo.

Ejecución de los ejemplos de Hello World

Puede ejecutar dos ejemplos de "Hello World": HelloWorld y HelloCICSWorld. El ejemplo HelloWorld utiliza solo los servicios Java y HelloCICSWorld enseña el uso de la clase TerminalPrincipalFacility de JCICS.

Antes de empezar

Asegúrese de que la región de CICS está configurada como se describe en "Ejecución de los ejemplos de JCICS" en la página 39.

Acerca de esta tarea

Los programas los inician las transacciones CICS de ejemplo. Los ejemplos utilizan las siguientes clases Java y programas CICS:

Ejemplo	Transacción	Programa	Clase Java
HelloWorld	JHE1	DFJ$JHE1	`HelloWorld`
		DFH$JSAM (programa C)	N/A
HelloCICSWorld	JHE2	DFJ$JHE2	`HelloCICSWorld`

DFH$JSAM es un programa CICS estándar que podría estar escrito en cualquiera de los lenguajes soportados por CICS. Por ejemplo, si no dispone de un compilador de C, podría escribir una versión COBOL de DFH$JSAM y utilizarla en lugar de la versión C provista. Como alternativa, puede eludir completamente DFH$JSAM modificando la definición de TRANSACTION JHE1 para que ejecute el programa DFJ$JHE1. No obstante, si cambia la definición, el programa Java no graba nada en el terminal, por lo que la única indicación de que la aplicación se ha ejecutado correctamente es el mensaje en el archivo `stdout`.

Procedimiento

- Especifique la transacción JHE1 en un terminal para ejecutar la aplicación Java estándar. Se reciben los siguientes mensajes de JHE1: El siguiente mensaje aparece en el terminal:

  ```
  SAMPLE *COMPLETED*, SEE STOUT
  ```

  La siguiente entrada se graba en el archivo `stdout`:

```
Hello from a regular Java application
```

- Especifique la transacción JHE2 en un terminal para ejecutar la aplicación JCICS. Se reciben los siguientes mensajes de JHE2 en el terminal:

```
Hello from a Java CICS application
```

Resultados

Ha ejecutado correctamente los ejemplos Hello World.

Qué hacer a continuación

Puede ejecutar otros ejemplos para probar distintos servicios que están disponibles para los programas Java en CICS.

Ejecución de ejemplos de control de programa

Puede ejecutar los ejemplos de canal y de COMMAREA para comprender cómo CICS procesa canales y contenedores o COMMAREA. Los programas pueden utilizar cualquier método para pasar datos, pero los contenedores no están limitados a 32 KB.

Antes de empezar

Asegúrese de que la región de CICS está configurada como se describe en "Ejecución de los ejemplos de JCICS" en la página 39.

Acerca de esta tarea

Los ejemplos enseñan cómo utilizar la clase `Program` de JCICS para pasar un canal y un contenedor o una COMMAREA a otro programa. El ejemplo de COMMAREA también enseña cómo convertir los caracteres ASCII que hay en el código Java a los equivalentes de EBCDIC (y desde este) utilizados por el programa de CICS nativo.

Los programas los inician las transacciones CICS de ejemplo. Los ejemplos utilizan las siguientes clases Java y programas CICS:

Ejemplo	Transacción	Programa	Clase Java
Canal	JPC3	DFJ$JPC3	`ProgramControl. .ClassThree`
		DFJ$JPC4	`ProgramControl. .ClassFour`
		DFH$LCCC (lenguaje C)	N/A
COMMAREA	JPC1	DFJ$JPC1	`ProgramControl. .ClassOne`
		DFJ$JPC2	`ProgramControl. .ClassTwo`
		DFH$LCCA (lenguaje C)	N/A

DFH$LCCA y DFH$LCCC son programas de CICS estándar que pueden estar escritos en cualquiera de los lenguajes de alto nivel soportados. Si no dispone de un compilador de C, podría escribir versiones COBOL de DFH$LCCA y DFH$LCCC y utilizarlas en lugar de las versiones C provistas.

Procedimiento

- Para el ejemplo de canal:
  1. Especifique la transacción JPC3 en un terminal. Se reciben los siguientes mensajes en `Task.out` (normalmente es su terminal):

     ```
     Entering ProgramControlClassThree.main()
     About to link to C program
     Leaving ProgramControlClassThree.main()
     ```

  2. Borre la pantalla. Se visualizan los siguientes mensajes:

     ```
     Entering ProgramControlClassFour.main()
     ProgramControlClassFour invoked with Container "IntData         "
     ProgramControlClassFour invoked with Container "StringData      "
     ProgramControlClassFour invoked with Container "Response        "
     Leaving ProgramControlClassFour.main()
     ```

     Los mensajes que indican los contenedores pueden aparecer en un orden distinto.

  El siguiente proceso tiene lugar en CICS:

  1. La transacción ejecuta la clase Java principal que se define en el recurso PROGRAM DFJ$JPC3. El programa Java construye un objeto Channel con dos contenedores y enlaza con el programa C, DFH$LCCC.
  2. DFH$LCCC procesa los contenedores, crea un nuevo contenedor de respuesta y lo devuelve.
  3. El programa Java comprueba los datos del contenedor de respuesta y planifica el inicio de una transacción pseudoconversacional pasando el objeto Channel a la transacción iniciada.
  4. La transacción iniciada ejecuta otra clase Java que se define en el recurso PROGRAM DFJ$JPC4. Este programa examina el `Channel` utilizando un objeto `ContainerIterator` y visualiza el nombre de cada contenedor que encuentra.

- Para el ejemplo de COMMAREA:
  1. Especifique la transacción de CICS JPC1 para ejecutar el ejemplo. Se reciben los siguientes mensajes en `Task.out` (normalmente es su terminal):

     ```
     Entering ProgramControlClassOne.main()
     About to link to C program
     Leaving ProgramControlClassOne.main()
     ```

  2. Borre la pantalla. Se visualizan los siguientes mensajes:

     ```
     Entering ProgramControlClassTwo.main()
     data received correctly
     Leaving ProgramControlClassTwo.main()
     ```

  El siguiente proceso tiene lugar en CICS:

  1. La transacción ejecuta la clase Java principal que se define en el recurso PROGRAM DFJ$JPC1. El programa Java construye una COMMAREA y enlaza con el programa C, DFH$LCCA.
  2. El programa C procesa la COMMAREA, la actualiza y la devuelve al programa Java.
  3. El programa Java comprueba los datos de la COMMAREA y planifica el inicio de una transacción pseudoconversacional pasando a la transacción iniciada los datos cambiados en su COMMAREA.
  4. La transacción iniciada ejecuta otra clase Java principal que se define en el recurso PROGRAM DFJ$JPC2. Este programa Java lee la COMMAREA y la vuelve a validar.

Ejecución del ejemplo TDQ

Puede ejecutar el ejemplo de la cola de datos transitoria para comprender cómo los programas Java pueden interaccionar con datos transitorios. Los programas pueden leer y grabar datos transitorios que estén almacenados en colas secuenciales.

Antes de empezar

Asegúrese de que la región de CICS está configurada como se describe en "Ejecución de los ejemplos de JCICS" en la página 39.

Acerca de esta tarea

Este ejemplo enseña cómo utilizar la clase TDQ de JCICS. El ejemplo utiliza las siguientes clase y programa Java:

Transacción	Programa	Clase Java
JTD1	DFJ$JTD1	`TDQ.ClassOne`

Procedimiento

Especifique la transacción JTD1 en un terminal para ejecutar el ejemplo. Se reciben los siguientes mensajes en `Task.out`:

```
Entering examples.TDQ.ClassOne.main()
Entering writeFixedData()
Leaving writeFixedData()
Entering writeFixedData()
Leaving writeFixedData()
Entering readFixedData()
Leaving readFixedData()
Entering readFixedDataConditional()
Leaving readFixedDataConditional()
Leaving examples.TDQ.ClassOne.main()
```

Resultados

CICS realiza el siguiente proceso:

1. La transacción ejecuta la clase Java principal que se define en el recurso PROGRAM DFJ$JTD1.
2. El programa Java graba algunos datos en una cola de datos transitoria, lee dicha cola y luego la suprime.

Ejecución del ejemplo de TSQ

Puede ejecutar el ejemplo de almacenamiento temporal para comprender cómo los programas Java pueden interaccionar con colas de almacenamiento temporal. Una cola de almacenamiento temporal es una cola de elementos de datos que se pueden leer y volver a leer en cualquier secuencia. La cola la crea una tarea y persiste hasta que la misma tarea u otra distinta la suprime.

Antes de empezar

Asegúrese de que la región de CICS está configurada como se describe en "Ejecución de los ejemplos de JCICS" en la página 39.

Acerca de esta tarea

Este ejemplo enseña cómo utilizar la clase TSQ de JCICS y cómo compilar una clase como una biblioteca de enlace dinámico (DLL) que se puede compartir con otros programas Java. Este ejemplo utiliza las siguientes clases y programas Java:

Transacción	Programa	Clase Java
JTS1	DFJ$JTS1	TSQ.ClassOne
	DFJ$JTSC	TSQ.Common

Procedimiento

Especifique la transacción de CICS JTS1 para ejecutar el ejemplo. Se reciben los siguientes mensajes en Task.out:

```
Entering TSQ.ClassOne.main()
Entering TSQ_Common.writeFixedData()
Leaving TSQ_Common.writeFixedData()
Entering TSQ_Common.serializeObject()
Leaving TSQ_Common.serializeObject()
Entering TSQ_Common.updateFixedData()
Leaving TSQ_Common.updateFixedData()
Entering TSQ_Common.writeConditionalFixedData()
Leaving TSQ_Common.writeConditionalFixedData()
Entering TSQ_Common.updateConditionalFixedData()
Leaving TSQ_Common.updateConditionalFixedData()
Entering TSQ_Common.readFixedData()
Leaving TSQ_Common.readFixedData()
Entering TSQ_Common.deserializeObject()
Leaving TSQ_Common.deserializeObject()
Entering TSQ_Common.readFixedConditionalData()
Number of items returned is 3
Leaving TSQ_Common.readFixedConditionalData()
Entering TSQ_Common.deleteQueue()
Leaving TSQ_Common.deleteQueue()
Leaving TSQ.ClassOne.main()
```

Resultados

El siguiente proceso tiene lugar en CICS:

1. La transacción ejecuta la clase Java principal que se define en el recurso PROGRAM DFJ$JTS1. El programa Java enlaza con otro programa Java común que se define en el recurso PROGRAM DFJ$JTSC.
2. El programa Java común graba en una cola de almacenamiento temporal auxiliar, actualiza la cola, suprime la cola y la devuelve.

Ejecución del ejemplo web

Puede ejecutar el ejemplo web para comprender cómo los programas Java pueden utilizar el soporte web de CICS para interaccionar con navegadores web.

Antes de empezar

Asegúrese de que la región de CICS está configurada como se describe en "Ejecución de los ejemplos de JCICS" en la página 39. Antes de ejecutar el programa de ejemplo web, siga las instrucciones en Configuración de los componentes de soporte web de CICS en la Guía de Internet. Utilice los programas de ejemplo DFH$WB1A (ensamblador) o DFH$WB1C (C) para confirmar que el soporte web de CICS está configurado correctamente.

Acerca de esta tarea

Este ejemplo enseña cómo utilizar las clases de documento y web de JCICS. A esta aplicación de ejemplo se accede desde un navegador web. El ejemplo obtiene información sobre la solicitud de cliente de entrada, las cabeceras HTTP y las características TCP/IP de la transacción. Esta información se graba en la secuencia de salida estándar `System.out` y se inserta en un documento de respuestas. También se obtiene información sobre el documento, que se graba en `System.out` y se inserta en el documento de respuestas. Luego el documento de respuestas se envía al cliente.

El ejemplo utiliza las siguientes clase y programa Java:

Program	Clase Java
DFJ$JWB1	Web.Sample1

Procedimiento

1. Lance el navegador web y escriba un URL que conecte con CICS mediante la vía de acceso absoluta `/CICS/CWBA/DFJ$JWB1` CICS devuelve el siguiente documento de respuestas al navegador web:

```
Web Sample1

Inbound Client Request Information:

Method: GET

Version: HTTP/1.1

Path: /cics/cwba/jcicxsa1

Request Type: HTTPYES

Query String: null

HTTP headers:

Value for HTTP header User-Agent is 'Mozilla/4.75 €en€ (WinNT; U)'

Browse of HTTP Headers started

Name: Host Value: winmvs2d.hursley.ibm.com:27361

Name: Connection Value: Keep-Alive, TE

Name: Accept Value: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png,
      */*

Name: Accept-Encoding Value: gzip

Name: Accept-Language Value: en

Name: Accept-Charset Value: iso-8859-1,*,utf-8

Name: Cookie Value: PBC_NLSP=en_US

Name: TE Value: chunked

Name: Via Value: HTTP/1.0 sp15ce18.hursley.ibm.com (IBM-PROXY-WTE-US)

Name: User-Agent Value: Mozilla/4.75 €en€ (WinNT; U)
```

```
Browse of HTTP Headers completed

TCPIP Information:

Client Name: sp15ce18.hursley.ibm.com

Server Name: winmvs2d.hursley.ibm.com

Client Address: 9.20.136.28

ClientAddrNu: 9.20.136.28

Server Address: 9.20.101.8

ServerAddrNu: 9.20.101.8

Clientauth: NO

SSL: NO

TcpipService: HTTPNSSL

PortNumber: 27361

Document Information:

Doctoken: 33 92 112 0 0 0 0 1 64 64 64 64 64 64 64 64

Docsize: 2762
```

2. Compruebe la secuencia de salida estándar en zFS. El ejemplo graba mensajes informativos en la secuencia de salida estándar `System.out` y mensajes de error en la secuencia de salida estándar `System.err`. Aquí tiene un ejemplo de la salida que se graba en la secuencia de salida `System.out`:

```
Sample1 started
Method: GET (3)
Version: HTTP/1.1 (8)
Path: /cics/cwba/jcicxsa1 (19)
Request Type: HTTPYES
Value for HTTP header User-Agent is 'Mozilla/4.75  en  (WinNT; U)'
HTTP headers:
Name: Host (4)
Value: winmvs2d.hursley.ibm.com:27361 (30)
Name: Connection (10)
Value: Keep-Alive, TE (14)
Name: Accept (6)
Value: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */* (67)
Name: Accept-Encoding (15)
Value: gzip (4)
Name: Accept-Language (15)
Value: en (2)
Name: Accept-Charset (14)
Value: iso-8859-1,*,utf-8 (18)
Name: Cookie (6)
Value: PBC_NLSP=en_US (14)
Name: TE (2)
Value: chunked (7)
Name: Via (3)
Value: HTTP/1.0 sp15ce18.hursley.ibm.com (IBM-PROXY-WTE-US) (52)
Name: User-Agent (10)
Value: Mozilla/4.75  en  (WinNT; U) (28)
Client Name: sp15ce18.hursley.ibm.com (24)
Server Name: winmvs2d.hursley.ibm.com (24)
Client Address: 9.20.136.28 (11)
ClientAddrNu: 9.20.136.28
Server Address: 9.20.101.8 (10)
ServerAddrNu: 9.20.101.8
```

```
Clientauth: NO
SSL: NO
TcpipService: HTTPNSSL
PortNumber: 27361
Doctoken: Doctoken: 33 92 112 0 0 0 0 1 64 64 64 64 64 64 64 64
Docsize: 2762
Sample1 complete
```

Desarrollo de aplicaciones utilizando el CICS Explorer SDK

El CICS Explorer Software Development Kit (SDK) proporciona un entorno para el desarrollo y el despliegue de aplicaciones Java en CICS, incluyendo el soporte de OSGi.

Acerca de esta tarea

Puede utilizar el SDK para crear aplicaciones nuevas o para volver a empaquetar aplicaciones Java existentes que cumplan con la especificación OSGi. La Plataforma de servicios OSGi proporciona un mecanismo para desarrollar aplicaciones utilizando un modelo de componentes y desplegar dichas aplicaciones en una infraestructura como paquetes OSGi. Un *paquete OSGi* es la unidad de despliegue para una aplicación y contiene información de control de versiones, dependencias y código de aplicación. El principal beneficio de OSGi es que se pueden crear aplicaciones a partir de componentes reutilizables a los que se accede únicamente mediante interfaces bien definidas denominadas *servicios OSGi*. También se pueden gestionar el ciclo de vida y las dependencias de aplicaciones Java de un modo granular. Para obtener información acerca del desarrollo de las aplicaciones con OSGi, consulte el sitio web OSGi Alliance.

Puede utilizar el SDK para desarrollar una aplicación Java para ejecutar en cualquier versión soportada de CICS. Diferentes versiones de CICS soportan distintas versiones de Java y la API de JCICS también se ha ampliado en versiones posteriores para el soporte de funciones adicionales de CICS. Para evitar la utilización de clases erróneas, el SDK brinda una función para configurar una plataforma de destino. Puede definir para qué versión de CICS está desarrollando y el SDK oculta automáticamente las clases Java que no puede utilizar.

Consulte *CICS Java Developer Guide* en la ayuda del SDK para obtener información detallada acerca de cómo realizar los siguientes pasos para desarrollar y desplegar las aplicaciones.

Procedimiento

1. Configure una plataforma de destino para el desarrollo de Java. La plataforma de destino asegura que únicamente se utilizan las clases Java que son adecuadas para las versiones destino de CICS en el desarrollo de la aplicación.
2. Cree un proyecto de plug-in para el desarrollo de la aplicación Java.
3. Desarrolle la aplicación Java utilizando las prácticas recomendadas. Si está dando sus primeros pasos en el desarrollo de aplicaciones Java para CICS, puede utilizar los ejemplos de JCICS que se proporcionan con el CICS Explorer SDK para comenzar. Para utilizar JCICS en una aplicación Java, debe importar el paquete com.ibm.cics.server.
4. Despliegue la aplicación Java en un paquete de CICS para zFS. Los paquetes CICS pueden contener uno o más paquetes OSGi y son la unidad de despliegue para la aplicación en CICS. Si ejecuta la aplicación Java en un servidor de JVM, debe conocer el nombre del recurso JVMSERVER en el que desea desplegar la aplicación.

Resultados

Ha desarrollado y desplegado correctamente la aplicación en un paquete CICS utilizando el CICS Explorer SDK.

Qué hacer a continuación

Cree un recurso BUNDLE de CICS para instalar los paquetes OSGi en un servidor de JVM. Si no puede crear recursos en la región CICS, el programador del sistema puede crear el recurso BUNDLE. Debe indicar al programador del sistema dónde está ubicado el directorio del paquete en zFS y el nombre del servidor de JVM de destino. Para obtener más detalles, consulte "Instalación de paquetes OSGi en un servidor de JVM" en la página 90.

Migración de aplicaciones utilizando el CICS Explorer SDK

Si tiene aplicaciones que se ejecutan en JVM agrupadas y desea ejecutarlas en un servidor de JVM, puede utilizar el CICS Explorer SDK para volver a empaquetar las aplicaciones como paquetes OSGi.

Acerca de esta tarea

Puede utilizar tres métodos para volver a empaquetar una aplicación Java existente. Cada método se explica de forma detallada en la ayuda relativa al SDK y se resume en el procedimiento que se indica a continuación.

Procedimiento

1. Compruebe que la aplicación Java es de enhebramiento seguro. Como el servidor de JVM es un entorno de ejecución multihebra, es importante que cualquier aplicación Java que se ejecute en ese entorno sea de enhebramiento seguro.
2. Compruebe que la aplicación Java no utiliza el método Java System.exit(). Si se utiliza este método, tanto el servidor de JVM como CICS se apagan.
3. Empaquete la aplicación Java como uno o más paquetes OSGi. Puede utilizar tres métodos para empaquetar la aplicación:

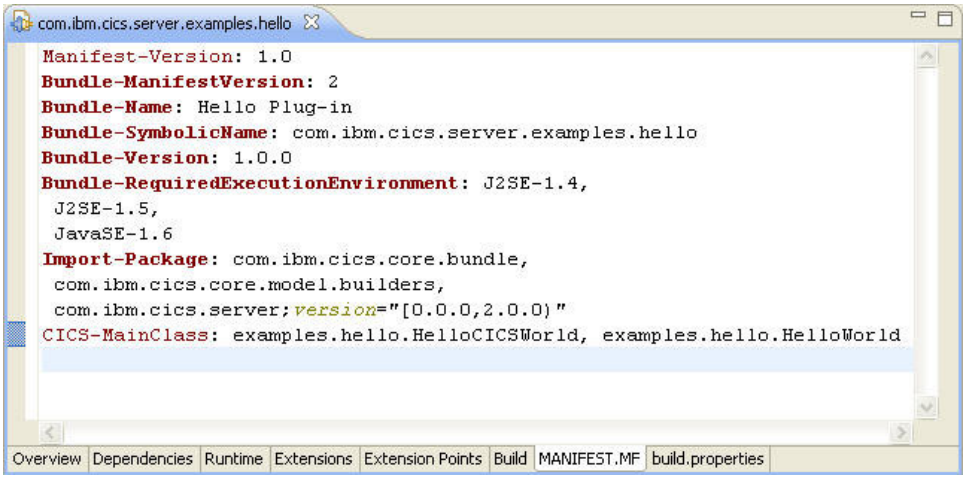
   Conversión
   Si ya tiene un proyecto Java de Eclipse para la aplicación Java y convierte dicho proyecto en un proyecto de plug-in OSGi. Este método es la práctica recomendada preferida. El paquete OSGi se puede ejecutar en un entorno de JVM en agrupación y en un servidor de JVM.

   Inyección
   Cree un proyecto de plug-in OSGi e importe el contenido del archivo JAR existente. Este método resulta útil si la aplicación ya es de enhebramiento seguro y no se necesitan refactorización ni volver a compilar. El paquete OSGi se puede ejecutar en un entorno de JVM en agrupación y en un servidor de JVM.

   Derivación
   Cree un proyecto de plug-in OSGi e importe un archivo JAR binario existente. Este método resulta útil en situaciones en las que hay restricciones de licencia o si no se puede extraer el archivo binario. Sin embargo, un paquete OSGi que contenga un archivo JAR no se soporta en un entorno de JVM en agrupación.

4. En cada uno de estos métodos, añada la declaración CICS-MainClass al manifiesto del proyecto.

   En la siguiente captura de pantalla, se muestra un ejemplo de archivo manifiesto para los ejemplos Hello de CICS. La aplicación de ejemplo contiene dos clases: HelloCICSWorld y HelloWorld, y estas se declaran en el archivo manifiesto de la declaración CICS-MainClass. Debe añadir una declaración CICS-MainClass a cada una de las clases utilizadas en la aplicación.



5. Despliegue los paquetes OSGi en un paquete de CICS para zFS. Debe especificar el recurso JVMSERVER de destino en el archivo de recursos de plug-in del paquete CICS.

Resultados

Tendrá una aplicación de enhebramiento seguro que se ha empaquetado como uno o más paquetes OSGi y que se despliega como un paquete de CICS en zFS.

Qué hacer a continuación

El programador del sistema puede crear los recursos necesarios para ejecutar la aplicación en un servidor de JVM, como se describe en “Movimiento de aplicaciones a un servidor de JVM” en la página 136.

Prácticas recomendadas para el desarrollo de aplicaciones Java en CICS

Cuando esté diseñando y desarrollando aplicaciones Java para ejecutarlas en CICS, asegúrese de que la aplicación no deja ningún estado no deseado en la JVM ni cambia el estado de la JVM de una forma no deseada. Puede utilizar servicios de CICS para ayudarse a controlar el estado de la JVM.

Aunque los servidores de JVM y las JVM en agrupación operan de formas distintas, cualquier aplicación Java que desarrolle puede seguir las mismas prácticas recomendadas para funcionar correctamente en ambos entornos de ejecución.

Protección del estado de una JVM

Si su aplicación modifica el estado de la JVM, asegúrese de que dicha aplicación también la restablece al estado original. Por ejemplo, una aplicación puede

restablecer el huso horario predeterminado y realizar cálculos basados en dicho huso horario. Otras aplicaciones que utilicen la misma JVM utilizan el nuevo huso horario predeterminado, que puede no ser adecuado.

- Si una aplicación se ejecuta en una JVM en agrupación, está aislada de otras aplicaciones. Sin embargo, las JVM en agrupación se reutilizan en serie y los cambios realizados por una aplicación pueden afectar a otras aplicaciones que se ejecuten en la misma JVM posteriormente.
- Si una aplicación se ejecuta en un servidor de JVM, no está aislada de otras aplicaciones que puedan estar ejecutándose también en la misma JVM en hebras distintas. Cualquier cambio que una aplicación realiza en la JVM afecta a las demás aplicaciones.

  No utilice métodos System.exit() en sus aplicaciones. El uso de los métodos System.exit() hace que el servidor de JVM y CICS se cierren y puede afectar al estado de las aplicaciones.

Control del estado estático de una JVM

No deje ningún estado no deseado en una JVM. El estado se pasa a las aplicaciones siguientes que utilicen la misma JVM en agrupación y, en un servidor de JVM, se comparte entre todas las aplicaciones en ejecución.

Una aplicación debe reinicializar su propio almacenamiento estático, si este depende del estado de un campo de clase que se puede cambiar. Los valores de las variables estáticas persisten en la JVM para todas las clases de aplicación y de sistema, incluidas las clases que pueden afectar a la aplicación, pero que esta no utiliza de forma explícita y los valores utilizados en los inicializadores estáticos.

En la mayoría de los casos, las variables estáticas se utilizan para evitar la reinicialización del almacenamiento y permitirles que persistan puede mejorar el rendimiento. Si la aplicación necesita que se restablezca el valor de estas variables, debe restablecer el valor por sí misma. Intente identificar y eliminar cualquier campo de clase modificable e inicializador estático que no se hayan incluido deliberadamente como parte del diseño de la aplicación.

Defina un campo de clase como private (privado) y final siempre que sea posible. Un método native (nativo) puede grabar en un campo de clase final, y un método que no sea private (privado) puede obtener el objeto referenciado por el campo de clase y puede modificar el estado del objeto o de la matriz.

Puede aprovechar la capacidad para pasar un estado a la hora de diseñar sus aplicaciones Java si desea que alguna información persista de una invocación de programa a la siguiente. El estado estático y las instancias de objetos referenciadas mediante estado estático persisten en la JVM, de forma que las aplicaciones tienen permiso para crear elementos persistentes que puedan utilizarse para ejecuciones futuras de la misma aplicación en la misma JVM.

Por ejemplo, una operación lee información de DB2 para construir una estructura de datos compleja; esto puede ser una operación cara que no desea repetir más veces de lo absolutamente necesario. La estructura de datos compleja se puede guardar en el almacenamiento estático de la aplicación y ser accesible para ejecuciones posteriores de la aplicación en la misma JVM, lo que evita una inicialización innecesaria. Si los objetos están anclados en el almacenamiento estático, es decir, en los campos de clase estáticos, nunca son candidatos para la recogida de basura.

- En un servidor de JVM, el estado estático persiste para todas las aplicaciones hasta que el servidor de JVM es inhabilitado por el programador del sistema. Puede proporcionar clases de activadores de paquetes OSGi para mantener el estado de los objetos en los reinicios del servidor de JVM. Estas clases no pueden contener llamadas de JCICS.
- En una JVM en agrupación, no hay garantía de que invocaciones posteriores de una aplicación se ejecutarán en la misma JVM. La aplicación no debe depender de la presencia de los elementos persistentes que se crean en la JVM. La aplicación puede comprobar su presencia para evitar cualquier inicialización innecesaria, pero debe estar preparada para inicializarlos si no se encuentran en la JVM actual.

Cierre de conexiones con DB2, sockets y otros recursos de sistema del tiempo de vida de la tarea tras su utilización

Si la aplicación se está ejecutando en una JVM agrupada, debe cerrar la conexión después de acceder a DB2. Si la aplicación no cierra la conexión, una ejecución posterior de la misma aplicación no podrá abrir dicha conexión. Si la aplicación se está ejecutando en un servidor de JVM, resulta posible tener varias conexiones a DB2 procedentes de distintas aplicaciones. Por lo tanto, cuando una tarea haya finalizado con DB2, se recomienda, aunque no es obligatorio, cerrar la conexión, porque esta se elimina cuando se completa la tarea.

Si se lanzan hebras en una aplicación para gestionar sockets utilizando el paquete java.net, la aplicación debe gestionar las conexiones y cerrarlas. Los sockets creados utilizando las clases java.net utilizan la prestación de sockets nativos en la JVM, en lugar del dominio de sockets de CICS. CICS no puede gestionar ni supervisar las comunicaciones que se realizan utilizando estos sockets.

Esto mismo se aplica a cualquier otro recurso de sistema del tiempo de vida de la tarea que utilice la aplicación, que debe liberarse tras su utilización.

Prueba de aplicaciones por si hay problemas de enhebramiento seguro

Escriba siempre aplicaciones Java de enhebramiento seguro. Puede utilizar la CICS JVM Application Isolation Utility para auditar la utilización de variables estáticas en sus aplicaciones Java. El programa de utilidad inspecciona los códigos de bytes de Java e informa sobre las variables estáticas que utiliza cada clase. Puede utilizar esta información para ayudarse a comprobar el código fuente. Asegúrese de que la aplicación restablece la variable estática correctamente en cada caso.

Si una aplicación Java funciona correctamente en su primer uso en una JVM concreta pero no se comporta adecuadamente en usos posteriores, es probable que el problema se deba a problemas de seguridad de hebra. En este caso, utilice el CICS JVM Application Isolation Utility como parte del trabajo de determinación de problemas para ayudarse a identificar el problema.

Interacción con datos estructurados procedentes de Java

Los programas Java de CICS a menudo interaccionan con datos que originalmente se diseñaron para su utilización con otros lenguajes de programación. Por ejemplo, un programa Java puede enlazar con un programa COBOL mediante una COMMAREA definida en un libro de copias COBOL o leer un registro desde un

archivo VSAM en el que los datos estén definidos mediante un archivo de cabecera de C++. Puede utilizar un importador para interaccionar con estas formas de datos estructurados.

Importación de datos de aplicación en Java utilizando JZOS y J2C

CICS soporta importadores de libro de copias, de forma que pueda utilizar datos estructurados de otros lenguajes de programación en Java. Los importadores soportados los proporcionan herramientas de JZOS y Rational, mediante la Java EE Connector Architecture (JCA), conocida también como la J2EE Connector Architecture (J2C).

Los importadores correlacionan los tipo de datos que contiene el programa fuente, de manera que su aplicación pueda acceder a campos individuales en estructuras de datos. Puede utilizar las herramientas J2C de JZOS o de Rational para interaccionar con datos y producir una clase Java, de forma que pueda pasar datos entre Java y otros programas en CICS.

CICS soporta artefactos Java de los siguientes importadores:

- Beans de enlace de datos de las herramientas J2C de Rational Application Developer (RAD) y Rational Developer for System z
- Registros del IBM JZOS Batch Toolkit para z/OS SDK

El IBM Redbook Java Application Development for CICS utiliza una aplicación de ejemplo llamada Heritage Trader que manipula una aplicación COBOL existente. En los siguientes temas se proporciona información:

- Instrucciones para instalar JZOS y J2C
- Migración de la aplicación COBOL a JCICS
- Creación de una clase Java de enlace de datos para J2C
- Generación de una clase de derivador con JZOS
- Implementaciones de ejemplo para acceso a web, archivos y DB2 utilizando la API de JCICS

Requisitos de J2C

Puede crear artefactos de conector de J2EE que se pueden utilizar para crear aplicaciones empresariales. El asistente RAD J2C le ayuda a crear una clase o un conjunto de clases que se correlacionan con COBOL y con otras estructuras de datos de programas de aplicación.

Necesita RAD en una estación de trabajo Windows o Linux para utilizar el importador de Rational J2C.

Requisitos de JZOS

El IBM JZOS Batch Toolkit para z/OS SDK es un conjunto de herramientas que brindan prestaciones por lotes de Java en z/OS. JZOS incluye un lanzador para ejecutar aplicaciones Java directamente como trabajos por lotes o como tareas iniciadas, y un conjunto de métodos Java que permiten acceder a datos tradicionales de z/OS y servicios del sistema clave directamente desde aplicaciones Java.

JZOS soporta la generación automática de clases de registro desde libros de copias de COBOL y DSECT de ensamblador.

La descarga JZOS incluye las publicaciones *JZOS COBOL Record Generator User's Guide* y *JZOS Assembler Record Generator User's Guide* en formato PDF.

IBM Redbooks

- Java Application Development for CICS
- Java Connectors for CICS Featuring the J2EE Connector Architecture
- Java Stand-alone Applications on z/OS Volume 2

Información sobre J2C

- RAD: Connecting to enterprise information systems (EIS)
- RAD: COBOL Importer overview
- CICS Transaction Gateway Programming Guide

Información sobre JZOS

- JZOS Java Launcher and Toolkit Overview
- JZOS Batch Launcher and Toolkit Installation and User Guide

Programación Java utilizando JCICS

Puede escribir aplicaciones Java que utilicen las la biblioteca de clases Java de JCICS (JCICS) para acceder a servicios de CICS. JCICS es el equivalente de Java de la interfaz de programación de aplicaciones (API) **EXEC CICS** que se proporciona para otros lenguajes soportados por CICS, como, por ejemplo, COBOL.

Mediante JCICS, puede escribir aplicaciones Java que accedan a recursos de CICS e integrarlas con programas escritos en otros lenguajes. La mayoría de las funciones de la API **EXEC CICS** están soportadas. La biblioteca se proporciona en el archivo `com.ibm.cics.server.jar` con CICS y con el CICS Explorer SDK.

La biblioteca de clases de Java para CICS (JCICS)

La biblioteca de clases de Java para CICS (JCICS) soporta la mayoría de las funciones de los mandatos de la API **EXEC CICS**.

Las clases JCICS están completamente documentadas en el Javadoc que se genera desde las definiciones de clases. El Javadoc está disponible en JCICS Class Reference.

JavaBeans

Algunas de las clases de JCICS se pueden utilizar como JavaBeans, lo que significa que se pueden personalizar en una herramienta de desarrollo de aplicaciones como Eclipse, serializarse y manipularse utilizando la API de JavaBeans.

Los siguientes JavaBeans están disponibles en JCICS:

- Program
- ESDS
- KSDS
- RRDS
- TDQ

- TSQ
- AttachInitiator
- EnterRequest

Estos beans no definen ningún suceso, sino que constan de propiedades y métodos. Se pueden crear instancias de los mismos en tiempo de ejecución de una de las tres formas siguientes:

- Mediante la llamada al método new para la propia clase. Este es el método preferido.
- Mediante la llamada a Beans.instantiate() para el nombre de la clase, con los valores de propiedad definidos manualmente.
- Mediante la llamada a Beans.instantiate() para un archivo `.ser`, con los valores de propiedad definidos en el tiempo de diseño.

Si se elige alguna de las dos primeras opciones, los valores de propiedad, incluidos el nombre del recurso de CICS, se deben definir invocando los métodos set adecuados en el tiempo de ejecución.

Estructura de biblioteca

Cada componente de biblioteca de JCICS pertenece en una de estas cuatro categorías: Interfaces, Clases, Excepciones o Errores.

`Interfaces`
Se proporcionan algunas interfaces para definir conjuntos de constantes. Por ejemplo, la interfaz TerminalSendBits proporciona un conjunto de constantes que se pueden utilizar para construir una `java.util.BitSet`.

`Clases`
Las clases incluidas proporcionan la mayoría de las funciones de JCICS. La clase `API` es una clase abstracta que proporciona inicialización común para cada clase que corresponda a una parte de la API de CICS, excepto para terminaciones anómalas (ABEND) y excepciones. Por ejemplo, la clase `Task` proporciona un conjunto de métodos y variables que se corresponden con una tarea de CICS.

`Errores y excepciones`
El lenguaje Java define tanto excepciones y errores como subclases de la clase `Throwable`. JCICS define `CicsError` como una subclase de `Error`. `CicsError` es la superclase para todas las demás clases de error de CICS, que se utilizan para errores graves.

JCICS define `CicsException` como una subclase de `Exception`. `CicsException` es la superclase para todas las clases de excepción de CICS (incluidas las clases `CicsConditionException`, como `InvalidQueueIdException`, que representa la condición QIDERR de CICS).

Consulte “Manejo de errores y terminación anómala” en la página 58 para obtener más información.

Recursos de CICS

Los recursos de CICS, como los programas o las colas de almacenamiento temporal, se representan mediante instancias de la clase Java adecuada, identificada por los valores de distintas propiedades, como el nombre del recurso.

Los recursos para CICS deben definirse utilizando el CICS Explorer, la transacción CEDA o los CICSPlex SM BAS. Consulte la *Guía de definición de recurso de CICS* o el manual *Conceptos y planificación de CICSPlex System Manager* para obtener

información sobre cómo definir recursos de CICS. Es posible utilizar acceso remoto implícito definiendo un recurso de forma local para que apunte a un recurso remoto.

Argumentos para pasar datos

Puede pasar datos entre programas utilizando canales y contenedores o mediante un área de comunicación (COMMAREA).

Si utiliza una COMMAREA, el límite que se puede pasar es de 32 KB cada vez. Si utiliza un canal y contenedores, puede pasar más de 32 KB entre programas. La COMMAREA o el canal, así como cualquier otro parámetro, se pasan como argumentos a los métodos apropiados.

Muchos de los métodos están sobrecargados: es decir, tienen versiones distintas que toman, bien un número distinto de argumentos, o bien argumentos de un tipo distinto. Puede haber un método que no tenga argumentos o los argumentos mínimos obligatorios y otro que tenga todos los argumentos. Por ejemplo, la clase `Program` incluye los siguientes métodos link() en :

`link()`
Este método realiza un LINK simple sin utilizar ninguna COMMAREA para pasar datos y ninguna otra opción.

`link(com.ibm.cics.server.CommAreaHolder)`
Este método realiza un LINK simple, utilizando una COMMAREA para pasar datos, pero sin ninguna otra opción.

`link(com.ibm.cics.server.CommAreaHolder, int)`
Este método realiza un LINK distribuido, utilizando una COMMAREA para pasar datos y un valor de DATALENGTH y un valor de DATALENGTH para especificar la longitud de los datos en la COMMAREA.

`link(com.ibm.record.IByteBuffer)`
Este método realiza un LINK utilizando un objeto que implementar la interfaz IByteBuffer del Java Record Framework que se proporciona con VisualAge for Java.

`link(com.ibm.cics.server.Channel)`
Este método realiza un LINK utilizando un canal para pasar datos en uno o más contenedores.

Clases serializables

Las clases serializables son clases JCICS que pueden sobrevivir a un ciclo Pasivo/Activado

La siguiente lista muestra las clases serializables:

- AddressResource
- AttachInitiator
- CommAreaHolder
- EnterRequest
- ESDS
- File
- KeyedFile
- KSDS
- NameResource
- Program

- RemotableResource
- Resource
- RRDS
- StartRequest
- SynchronizationResource
- SyncLevel
- TDQ
- TSQ
- TSQType

System.out y System.err

Para cada tarea de CICS relacionada con Java, CICS crea automáticamente dos clases `PrintWriters` de Java que se pueden utilizar como secuencias de salida estándar y de error estándar. Las secuencias de salida estándar y de error estándar son campos públicos de la clase `Task` llamados `out` y `err`.

Si una tarea de CICS se controla desde un terminal (en este caso el terminal se llama *recurso principal*), CICS correlaciona las secuencias de salida estándar y de error estándar con el terminal de la tarea.

Si la tarea no tiene ningún terminal como recurso principal, las secuencias de salida estándar y de error estándar se envían a System.out y System.err. System.out y System.err se correlacionan, respectivamente, con las colas de datos transitorias CESO y CESE de CICS. El programador del sistema CICS crea estas colas y otras que se utilizan para mensajes de CICS durante la instalación de CICS. Puede acceder a estas colas de mensajes e imprimirlas o visualizarlas mediante programas de utilidad, como el programa de ejemplo DFH$TDWT. DFH$TDWT se encuentra en CICSTS42.CICS.SDFHLOAD.

Hebras

Solo la hebra inicial de la JVM puede acceder a la API de JCICS. Puede crear otras hebras, pero debe dirigir todas las solicitudes a la API de JCICS mediante la hebra inicial. En un entorno de servidor de JVM, varias hebras iniciales pueden acceder a la API de JCICS utilizando la misma JVM.

Además, debe asegurarse de que todas las hebras que no sean la hebra inicial hayan terminado antes de hacer cualquiera de las siguientes cosas:

- link()
- xctl()
- setNextTransaction(), setNextCOMMAREA()
- commit(), rollback()
- devolver una `AbendException`

Referencia de servicios de JCICS

Muchas de las opciones y de los servicios disponibles para programas que no son Java a través de la API **`EXEC CICS`** están disponibles para programas Java mediante JCICS.

Manejo de excepciones de CICS en programas Java

Las terminaciones anómalas (ABEND) y las excepciones de CICS se integran en la arquitectura de manejo de excepciones de Java para manejar los problemas que se producen en CICS.

Todas las ABEND normales de CICS se correlacionan con una única excepción Java, `AbendException`, mientras que cada condición de CICS se correlaciona con una excepción Java distinta. Esto conduce a un modelo de manejo de ABEND en Java que es similar a los demás lenguajes de programación; se proporciona control a un único manejador para cada ABEND, y el manejador debe consultar la ABEND concreta y luego decidir qué hacer.

Si el propio CICS detecta la excepción que representa una condición, se convierte en una ABEND.

El manejo de excepciones de Java está completamente integrado con el manejo de ABEND y de condiciones en otros lenguajes, de forma que las ABEND se puedan propagar entre Java y programas que no son de Java de la manera estándar independiente del lenguaje. Una condición se correlaciona con una ABEND antes de que salga del programa que provocó o detectó la condición.

Sin embargo, hay varias diferencias con el modelo de manejo de finalizaciones anómalas para otros lenguajes de programación, que se debe a la naturaleza de la arquitectura de manejo de excepciones de Java y a la implementación de parte de la tecnología subyacente a la API de Java:

- Las ABEND que no se pueden manejar en otros lenguajes de programación pueden detectarse en programas Java. Estas ABEND normalmente se producen durante el proceso de punto de sincronización. Para evitar que estas ABEND interrumpan aplicaciones Java, se correlacionan con una extensión de una excepción sin comprobar; por lo tanto, no tienen que declararse ni detectarse.
- Varios sucesos internos de CICS, como la terminación de programas, también se correlacionan con excepciones de Java y, por lo tanto, una aplicación Java puede detectarlos. De nuevo, para evitar interrumpir el caso normal, estos sucesos se correlacionan con extensiones de una excepción sin comprobar y no tienen que detectarse ni declararse.

Tres jerarquías de clases de excepciones están relacionadas con CICS:

1. CicsError, que amplía java.lang.Error y es la base para AbendError y UnknownCicsError.
2. CicsRuntimeException, que amplía java.lang.RuntimeException y, a su vez, es ampliada por:

   `AbendException`
   Representa una terminación anómala normal de CICS.

   `EndOfProgramException`
   Indica que un programa con el que está enlazado ha terminado normalmente.

   `TransferOfControlException`
   Indica que un programa ha utilizado un método xctl(), el equivalente del mandato XCTL de CICS.
3. CicsException, que amplía java.lang.Exception y tiene la subclase:

   `CicsConditionException.`
   La clase base para todas las condiciones de CICS.

Mandatos para manejo de errores de CICS:

El manejo de condiciones de CICS está integrado en la arquitectura de excepciones de Java, como se describe anteriormente. El modo en el que el mandato "**`EXEC CICS`**" equivalente está soportado se describe a continuación:

HANDLE ABEND
Para manejar una ABEND (terminación anómala) generada por un programa en cualquier lenguaje soportado por CICS, utilice una sentencia try-catch de Java, con AbendException en una cláusula catch.

HANDLE CONDITION
Para manejar una condición específica, como PGMIDERR, utilice una cláusula catch que indique la excepción adecuada; en este caso, InvalidProgramException. Como alternativa, utilice una cláusula catch que indique CicsConditionException, si hay que detectar todas las condiciones de CICS.

IGNORE CONDITION
Este mandato no es relevante en aplicaciones Java.

POP HANDLE y PUSH HANDLE
Estos mandatos no son relevantes en aplicaciones Java. Las excepciones de Java utilizadas para representar ABEND y condiciones de CICS las detecta cualquier bloque catch en el ámbito.

Condiciones de CICS:

El modelo de manejo de condiciones en Java es distinto al de otros lenguajes de programación de CICS.

En COBOL se puede definir un código de manejo de excepciones para cada condición. Si esa condición se produce durante el proceso de un mandato de CICS, el control se transfiere al código.

En C y C++ no es posible definir un código de manejo de excepciones para una condición; para detectar una condición, debe seleccionarse el campo RESP del EIB tras cada mandato de CICS.

En Java, cualquier condición devuelta por un mandato de CICS se correlaciona con una excepción de Java. Puede incluir todos los mandatos de CICS en un bloque try-catch y realizar proceso específico para cada condición, o contar con una única cláusula null catch si la excepción en concreto no es relevante. Como alternativa, puede permitir que la condición se propague para que la maneje una cláusula catch en un ámbito mayor.

Consulte “Correlación de excepciones de JCICS” en la página 75 para obtener una descripción de la relación entre las condiciones de CICS y las excepciones de Java.

Manejo de errores y terminación anómala

Para iniciar una ABEND (terminación anómala) desde un programa Java, debe invocar uno de los métodos Task.abend() o Task.forceAbend().

Métodos	Clase de JCICS	Mandatos EXEC CICS
abend(), forceAbend()	Task	ABEND

ABEND
Para iniciar una ABEND desde un programa Java, invoque uno de los métodos Task.abend(). Esto provoca que se defina una condición de terminación anómala en CICS y que se genere una AbendException. Si la AbendException no se detecta en un nivel superior del objeto de aplicación, o si la maneja un manejador ABEND registrado en el programa de llamada (si lo hay), CICS termina la transacción y la retrotrae.

Los distintos métodos abend() son:

- abend(String *abcode*), que provoca una ABEND con el código de ABEND *abcode*.
- abend(String *abcode*, boolean *dump*), que provoca una ABEND con el código de ABEND *abcode*. Si el parámetro **dump** es false, no se realiza volcado.
- abend(), que provoca una ABEND sin código de ABEND y sin volcado.

ABEND CANCEL

Para lanzar una ABEND que no se pueda manejar, invoque uno de los métodos Task.forceAbend(). Como se describe anteriormente, esto provoca que se genere una AbendCancelException que se puede detectar en programas Java. Si lo hace, debe volver a generar la excepción para completar el proceso de **ABEND_CANCEL**, de manera que, cuando el control regrese a CICS, CICS terminará y retrotraerá la transacción. Detecte solo la AbendCancelException para fines de notificación y luego vuelva a generarla.

Los distintos métodos forceAbend() son:

- forceAbend(String *abcode*), que provoca una **ABEND CANCEL** con el código de ABEND *abcode*.
- forceAbend(String *abcode*, boolean *dump*), que provoca una **ABEND CANCEL** con el código de ABEND *abcode*. Si el parámetro **dump** es false, no se realiza volcado.
- forceAbend(), que provoca una **ABEND CANCEL** sin código de ABEND y sin volcado.

Conversaciones correlacionadas de APPC

La API de JCICS no dispone de soporte para conversación no correlacionada de APPC.

Conversaciones correlacionadas de APPC:

Métodos	**Clase de JCICS**	**Mandatos de EXEC CICS**
initiate()	AttachInitiator	ALLOCATE, CONNECT PROCESS
converse()	Conversation	CONVERSE
Métodos get*()	Conversation	EXTRACT ATTRIBUTES
Métodos get*()	Conversation	EXTRACT PROCESS
free()	Conversation	FREE
issueAbend()	Conversation	ISSUE ABEND
issueConfirmation()	Conversation	ISSUE CONFIRMATION
issueError()	Conversation	ISSUE ERROR
issuePrepare()	Conversation	ISSUE PREPARE
issueSignal()	Conversation	ISSUE SIGNAL
receive()	Conversation	RECEIVE
send()	Conversation	SEND
flush()	Conversation	WAIT CONVID

Soporte de correlación básico (BMS)

El soporte de correlación básico (BMS) es una interfaz de programación de aplicaciones entre programas de CICS y dispositivos de terminal. JCICS proporciona soporte para algunas de las interfaces de programación de aplicaciones de BMS.

Métodos	Clase de JCICS	Mandatos de EXEC CICS
sendControl()	TerminalPrincipalFacility	SEND CONTROL
sendText()	TerminalPrincipalFacility	SEND Text
	No está soportado.	SEND MAP, RECEIVE MAP

Canales y contenedores

Los *contenedores* son bloques de datos con nombre diseñados para pasar información entre programas. Los contenedores se agrupan en conjuntos llamados *canales*. Puede utilizar mandatos de JCICS relacionados con canales y contenedores al escribir enterprise beans de CICS. Sin embargo, CICS no soporta la transmisión de canales sobre secuencias de solicitudes de IIOP.

Para obtener información introductoria sobre canales y contenedores, así como pautas sobre cómo utilizar canales en aplicaciones que no sean Java, consulte Transferencia de datos entre programas mejorada utilizando canales en programación de aplicaciones CICS.

Para obtener información sobre herramientas que permiten a los programas Java acceder a datos de aplicación de CICS existentes, consulte Interacción con datos estructurados de Java en Aplicaciones Java en CICS.

Nota: CICS no soporta la transmisión de canales sobre secuencias de solicitudes de IIOP y, por ejemplo, no se puede pasar un canal a un enterprise bean de una región remota.

Tabla 2 lista las clases y los métodos que implementan el soporte de JCICS para canales y contenedores.

Tabla 2. Soporte de JCICS para canales y contenedores

Métodos	Clase de JCICS	Mandatos de EXEC CICS
containerIterator()	Channel	STARTBROWSE CONTAINER
createContainer()	Canal	
deleteContainer()	Channel	DELETE CONTAINER CHANNEL
getContainer()	Channel	
getName()	Channel	
delete()	Container	DELETE CONTAINER CHANNEL
get(), getLength()	Container	GET CONTAINER CHANNEL [NODATA]
getName()	Container	
put()	Container	PUT CONTAINER CHANNEL
getOwner()	ContainerIterator	
hasNext()	ContainerIterator	
next()	ContainerIterator	GETNEXT CONTAINER BROWSETOKEN
remove()	ContainerIterator	
link()	Program	LINK
xctl()	Program	XCTL

Tabla 2. Soporte de JCICS para canales y contenedores (continuación)

Métodos	**Clase de JCICS**	**Mandatos de `EXEC CICS`**
setNextChannel()	TerminalPrincipalFacility	RETURN CHANNEL
issue()	StartRequest	START CHANNEL
createChannel()	Task	
getCurrentChannel()	Task	ASSIGN CHANNEL
containerIterator()	Task	STARTBROWSE CONTAINER

La condición de CICS CHANNELERR tiene como resultado la generación de una ChannelErrorException; la condición de CICS CONTAINERERR produce una ContainerErrorException; la condición de CICS CCSIDERR resulta en una CCSIDErrorException.

Creación de canales y contenedores en JCICS:

Para crear un canal, utilice el método createChannel() de la clase Task.

Por ejemplo:

```
Task t=Task.getTask();
Channel custData = t.createChannel("Datos_cliente");
```

La cadena proporcionada al método createChannel es el nombre por el que CICS conoce el objeto Channel. (El nombre se rellena con espacios hasta llegar a 16 caracteres, para cumplir las convenciones de nomenclatura de CICS).

Para crear un nuevo contenedor en el canal, utilice el método createContainer() de Canal. Por ejemplo:

```
Container custRec = custData.createContainer("Registro_cliente");
```

La cadena proporcionada al método createContainer es el nombre por el que CICS conoce el objeto Container. (El nombre se rellena con espacios hasta llegar a 16 caracteres, si es necesario, para cumplir las convenciones de nomenclatura de CICS). Si ya existe un contenedor con el mismo nombre en este canal, se genera una ContainerErrorException.

Colocación de datos en un contenedor:

Para poner datos en un objeto Container (contenedor), utilice el método Container.put().

Para poner datos en un objeto Container (contenedor), utilice el método Container.put(). Los datos se pueden añadir a un contenedor como una matriz de bytes o como una serie. Por ejemplo:

```
String custNo = "00054321";
byte[] custRecIn = custNo.getBytes();
custRec.put(custRecIn);
```

O:

```
custRec.put("00054321");
```

Paso de un canal a otro programa u otra tarea:

Para pasar un canal en una llamada de enlace de programa o de control de programa de transferencia (XCTL), utilice los métodos link() y xctl() de la clase Program respectivamente.

```
programX.link(custData);

programY.xctl(custData);
```

Para definir el siguiente canal en una llamada de devolución de programa, utilice el método setNextChannel() de la clase TerminalPrincipalFacility:

```
terminalPF.setNextChannel(custData);
```

Para pasar un canal en una solicitud START, utilice el método issue de la clase StartRequest:

```
startrequest.issue(custData);
```

Recepción del canal actual:

No es necesario que un programa reciba su canal actual explícitamente. Sin embargo, un programa puede obtener su canal actual de la tarea actual.

Si un programa obtiene el canal actual de la tarea actual, dicha tarea puede extraer los contenedores por nombre:

```
Task t = Task.getTask();
Channel custData = t.getCurrentChannel();
if (custData != null) {
    Container custRec = custData.getContainer("Registro_cliente");
} else {
    System.out.println("No hay canal actual");
}
```

Obtención de datos de un contenedor:

Utilice el método Container.get() para leer los datos de un contenedor que esté en una matriz de bytes.

```
byte[] custInfo = custRec.get();
```

Examen del canal actual:

Un programa JCICS al que se le pasa un canal puede acceder a todos los objetos `Container` sin recibir el canal explícitamente.

Para ello, utiliza un objeto `ContainerIterator`. (La clase `ContainerIterator` implementa la interfaz java.util.Iterator). Cuando se crea una instancia de un objeto Task a partir de la tarea actual, su método containerIterator() devuelve un Iterator para el canal actual o un valor nulo si no hay ningún canal actual. Por ejemplo:

```
Task t = Task.getTask();
ContainerIterator ci = t.containerIterator();
While (ci.hasNext()) {
    Container custData = ci.next();
    // Procese el contenedor...
}
```

Un ejemplo de JCICS:

Este ejemplo muestra un extracto de una clase Java denominada Payroll que llama a un programa de servidor COBOL denominado `PAYR`. La clase Payroll utiliza las

clases com.ibm.cics.server.Channel y com.ibm.cics.server.Container de JCICS para hacer lo mismo que un programa cliente que no fuera de Java haría mediante mandatos **EXEC CICS**.

```
import com.ibm.cics.server.*;
public class Payroll {
     ...
     Task t=Task.getTask();

     // Cree el canal payroll_2004 channel
     Channel payroll_2004 = t.createChannel("payroll-2004");

     // Cree el contenedor de empleado (employee)
     Container employee = payroll_2004.createContainer("empleado");

     // Ponga el nombre del empleado en el contenedor
     employee.put("John Doe");

     // Cree el contenedor de salario (wage)
     Container wage = payroll_2004.createContainer("salario");

     // Ponga el salario en el contenedor
     wage.put("2000");

     // Enlace con el programa PAYROLL pasando el canal payroll_2004
     Program p = new Program();
     p.setName("PAYR");
     p.link(payroll_2004);

     // Obtenga el contenedor de estado que se ha devuelto
     Container status = payroll_2004.getContainer("estado");

     // Obtenga la información de estado
     byte[] payrollStatus = status.get();
     ...
}
```

Figura 3. Clase Java que utiliza las clases JCICS com.ibm.cics.server.Channel y com.ibm.cics.server.Container para pasar un canal a un programa de servidor de COBOL

Servicios de diagnóstico

La interfaz de programación de aplicaciones de JCICS dispone de soporte para estos mandato de rastreo y volcado de CICS.

Métodos	Clase de JCICS	Mandatos de **EXEC CICS**
	No está soportada.	DUMP
enterTrace()	EnterRequest	ENTER
enableTrace(), disableTrace()	Region, Task	TRACE

Servicios de documento

Esta sección describe el soporte de JCICS para los mandatos de la interfaz de programación de aplicaciones DOCUMENT.

La clase `Document` correlaciona con la API **EXEC CICS DOCUMENT**. Los constructores de la clase `DocumentLocation` se correlacionan con las palabras clave AT y TO de la API **EXEC CICS DOCUMENT**. Los métodos de establecimiento y de obtención de la clase SymbolList se correlacionan con las palabras clave SYMBOLLIST, LENGTH, DELIMITER, y UNESCAPE de la API **EXEC CICS DOCUMENT**.

Métodos	Clase de JCICS	Mandatos de **EXEC CICS**
create*()	Document	DOCUMENT CREATE
append*()	Document	DOCUMENT INSERT
insert*()	Document	DOCUMENT INSERT
addSymbol()	Document	DOCUMENT SET
setSymbolList()	Document	DOCUMENT SET
retrieve*()	Document	DOCUMENT RETRIEVE
get*()	Document	DOCUMENT

Servicios de entorno

Los servicios de entorno de CICS proporcionan acceso a áreas de datos, parámetros y atributos de recurso de CICS que son relevantes para un programa de aplicación.

Los mandatos y las opciones de **EXEC CICS** que cuentan con soporte de JCICS equivalente son:

- ADDRESS
- ASSIGN
- INQUIRE SYSTEM
- INQUIRE TASK
- INQUIRE TERMINAL/NETNAME

ADDRESS:

Se proporciona el siguiente soporte para las opciones de mandato de API **ADDRESS**.

Para obtener información completa sobre el mandato **EXEC CICS ADDRESS**, consulte ADDRESS en programación de aplicaciones CICS.

ACEE
El elemento de entorno de descriptor de acceso (ACEE) lo crea un gestor de seguridad externo cuando un usuario de CICS inicia sesión. Esta opción no se soporta en JCICS.

COMMAREA
Una COMMAREA contiene datos de usuario que se pasan con un mandato. El puntero de COMMAREA se pasa automáticamente al programa enlazado mediante el argumento **CommAreaHolder**. Consulte "Argumentos para pasar datos" en la página 55 para obtener más información.

CWA El área de trabajo común (CWA) contiene datos de usuario globales, que se pueden compartir entre tareas.

EIB El contiene información sobre el último mandato de CICS ejecutado. El acceso a los valores del EIB lo proporcionan métodos en los objetos apropiados. Por ejemplo,

eibtrnid
lo devuelve el método getTransactionName() de la clase Task.

eibaid lo devuelve el método getAIDbyte() de la clase TerminalPrincipalFacility.

eibcposn
lo devuelven los métodos getRow() y getColumn() de la clase Cursor.

TCTUA
El área de usuario de la tabla de control de terminal (TCTUA) contiene datos de usuario asociados con el terminal que controla la transacción de CICS (el recurso principal). Esta área se utiliza para pasar información entre programas de aplicación, pero solo si el mismo terminal está asociado con los programas de aplicación implicados. El contenido del TCTUA se puede obtener utilizando el método getTCTUA() de la clase TerminalPrincipalFacility.

TWA El área de trabajo de transacción (TWA) contiene datos de usuario que están asociados con la tarea de CICS. Esta área se utiliza para pasar información entre programas de aplicación, pero solo si están en la misma tarea. Se puede obtener una copia de la TWA utilizando el método getTWA() de la clase Task.

ASSIGN:

Se proporciona el siguiente soporte para las opciones de mandato de API **ASSIGN**.

Para obtener información detallada sobre este mandato, consulte ASSIGN en programación de aplicaciones CICS.

Métodos	**Clase de JCICS**
getABCODE()	AbendException
getAPPLID()	Region
getCurrentChannel()	Task
getCWA()	Region
getName()	TerminalPrincipalFacility o ConversationPrincipalFacility
getFCI()	Task
getNetName()	TerminalPrincipalFacility o ConversationPrincipalFacility
getPrinSysid()	TerminalPrincipalFacility o ConversationPrincipalFacility
getProgramName()	Task
getQNAME()	Task
getSTARTCODE()	Task
getSysid()	Region
getTCTUA()	TerminalPrincipalFacility
getTERMCODE()	TerminalPrincipalFacility
getTWA()	Task
getUSERID(), Task.getUSERID()	Task, TerminalPrincipalFacility o ConversationPrincipalFacility

No hay soporte para ninguna otra opción de ASSIGN.

INQUIRE SYSTEM:

Se proporciona el siguiente soporte para las opciones de SPI **INQUIRE SYSTEM**.

Métodos	Clase de JCICS
getAPPLID()	Region
getSYSID()	Region

No hay soporte para ninguna otra opción de **INQUIRE SYSTEM**.

INQUIRE TASK:

Se proporciona el siguiente soporte para las opciones de mandato de API **INQUIRE TASK**.

Métodos	Clase de JCICS
getSTARTCODE()	Task
getTransactionName()	Task
getUSERID()	Task

FACILITY
Puede encontrar el nombre del recurso principal de la tarea llamando al método getName() en el recurso principal de la tarea, que, a su vez, se puede encontrar llamando al método getPrincipalFacility() en el objeto Task actual.

FACILITYTYPE
Puede determinar el tipo de recurso utilizando el operador instanceof de Java para comprobar la clase de la referencia de objeto devuelta.

No hay soporte para ninguna otra opción de INQUIRE TASK.

INQUIRE TERMINAL e INQUIRE NETNAME:

Se proporciona el siguiente soporte para las opciones de SPI **INQUIRE TERMINAL** e **INQUIRE NETNAME**.

Métodos	Clase de JCICS
getUSERID()	Terminal, ConversationalPrincipalFacility
Terminal.getUser()	Terminal, ConversationalPrincipalFacility

También puede encontrar el valor de USERID llamando al método getUSERID() en el objeto Task actual o en el objeto que representa el recurso principal de la tarea.

No hay soporte para ninguna otra opción de **INQUIRE TERMINAL** o **INQUIRE NETNAME**.

Servicios de archivos

JCICS proporciona clases y métodos que se correlacionan con los mandatos de la API de **EXEC CICS** para cada tipo de archivo e índice de CICS.

Para obtener información sobre herramientas que permiten a los programas Java acceder a datos de aplicación de CICS existentes, consulte Interacción con datos estructurados de Java en Aplicaciones Java en CICS.

CICS soporta los siguientes tipos de archivos:

- Conjunto de datos secuenciados de claves (KSDS)
- Conjunto de datos secuenciados de entrada (ESDS)
- Conjunto de datos de registro relativo (RRDS)

Los archivos KSDS y ESDS pueden contar con índices alternativos (o secundarios). CICS no soporta el acceso a un archivo RRDS mediante un índice secundario. CICS trata los índices secundarios como si fueran archivos KSDS independientes en sí mismos, lo que significa que cuentan con entradas de FD independientes.

Hay unas pocas diferencias entre el acceso a archivos KSDS, ESDS (índice primario) y ESDS (índice secundario), lo que significa que no siempre se puede utilizar una interfaz común.

Los registros se pueden leer, actualizar, suprimir y examinar en todos los tipos de archivo, con la excepción de registros que no se pueden suprimir desde un archivo ESDS.

Consulte Conjuntos de datos VSAM: KSDS, ESDS, RRDS para obtener más información sobre los conjuntos de datos.

Los mandatos de Java que leen datos solo soportan el equivalente de la opción SET en mandatos de **EXEC CICS**. Los datos devueltos se copian automáticamente desde almacenamiento de CICS a un objeto de Java.

Las interfaces de Java relacionadas con control de archivos están en cinco categorías:

File La superclase para otras clases de archivo; contiene métodos comunes para todas las clases de archivo.

KeyedFile
Contiene las interfaces comunes a un archivo KSDS al que se accede utilizando el índice primario, un archivo KSDS al que se accede utilizando un índice secundario y un archivo ESDS al que se accede utilizando un índice secundario.

KSDS Contiene la interfaz específica a los archivos KSDS.

ESDS Contiene la interfaz específica a archivos ESDS a los que se accede mediante dirección de byte relativa (RBA, su índice primario) o dirección de byte ampliada (XRBA). Para utilizar XRBA en lugar de RBA, emita el método setXRBA(true).

RRDS Contiene la interfaz específica a archivos RRDS a los que se accede mediante número relativo de registro (RRN, su índice primario).

Para cada archivo, ha dos objetos sobre los que se puede operar; el objeto File y el objeto FileBrowse. El objeto File representa el propio archivo y se puede utilizar con métodos para realizar las siguientes operaciones de API:

- DELETE
- READ
- REWRITE
- UNLOCK
- WRITE
- STARTBR

Un objeto File lo crea la aplicación de usuario explícitamente iniciando la clase de archivo necesaria. El objeto FileBrowse representa una operación de examen de un archivo. Puede haber más de un examen activo en un archivo específico en cualquier momento, y cada examen se distingue por un REQID. Se pueden crear instancias de métodos para que un objeto FileBrowse realice las siguientes operaciones de API:

- ENDBR
- READNEXT
- READPREV
- RESETBR

La aplicación de usuario no crea instancias de un objeto FileBrowse explícitamente; este se crea y se devuelve a la clase de usuario mediante los métodos que realizan la operación STARTBR.

Las siguientes tablas muestran cómo se correlacionan las clases y los métodos de JCICS con los mandatos de la API de **EXEC CICS** para cada tipo de archivo e índice de CICS. En estas tablas, las clases y los métodos de JCICS se muestran con la forma `clase.método()`. Por ejemplo, KeyedFile.read() hace referencia al método read() en la clase KeyedFile.

La primera tabla muestra las clases y los métodos para archivos con claves:

Tabla 3. Clases y métodos para archivos con claves

Clase de índice primario y secundario y método de KSDS	Clase de índice secundario y método de ESDS	Mandato de la API File de CICS
KeyedFile.read()	KeyedFile.read()	**READ**
KeyedFile.readForUpdate()	KeyedFile.readForUpdate()	**READ UPDATE**
KeyedFile.readGeneric()	KeyedFile.readGeneric()	**READ GENERIC**
KeyedFile.rewrite()	KeyedFile.rewrite()	**REWRITE**
KSDS.write()	KSDS.write()	**WRITE**
KSDS.delete()		**DELETE**
KSDS.deleteGeneric()		**DELETE GENERIC**
File.unlock()	File.unlock()	**UNLOCK**
KeyedFile.startBrowse()	KeyedFile.startBrowse()	**START BROWSE**
KeyedFile.startGenericBrowse()	KeyedFile.startGenericBrowse()	**START BROWSE GENERIC**
KeyedFileBrowse.next()	KeyedFileBrowse.next()	**READNEXT**
KeyedFileBrowse.previous()	KeyedFileBrowse.previous()	**READPREV**
KeyedFileBrowse.reset()	KeyedFileBrowse.reset()	**RESET BROWSE**
FileBrowse.end()	FileBrowse.end()	**END BROWSE**

Esta tabla muestra las clases y los métodos para archivos sin claves. A ESDS y RRDS se accede mediante sus índices primarios:

Clase de índice primario y método de ESDS	Clase de índice primario y método de RRDS	Mandato de la API File de CICS
ESDS.read()	RRDS.read()	**READ**
ESDS.readForUpdate()	RRDS.readForUpdate()	**READ UPDATE**

Clase de índice primario y método de ESDS	Clase de índice primario y método de RRDS	Mandato de la API File de CICS
ESDS.rewrite()	RRDS.rewrite()	**REWRITE**
ESDS.write()	RRDS.write()	**WRITE**
	RRDS.delete()	**DELETE**
File.unlock()	File.unlock()	**UNLOCK**
ESDS.startBrowse()	RRDS.startBrowse()	**START BROWSE**
ESDS_Browse.next()	RRDS_Browse.next()	**READNEXT**
ESDS_Browse.previous()	RRDS_Browse.previous()	**READPREV**
ESDS_Browse.reset()	RRDS_Browse.reset()	**RESET BROWSE**
FileBrowse.end()	FileBrowse.end()	**END BROWSE**
ESDS.setXRBA()		

Los datos que se deben grabar en un archivo deben estar en una matriz de bytes de Java.

Los datos se leen desde un archivo a un objeto RecordHolder, el almacenamiento lo proporciona CICS y se publican automáticamente al final del programa.

No es necesario especificar el valor de **KEYLENGTH** en cualquier método File; la longitud utilizada es la longitud real de la clave que se pasa. Cuando se crea un objeto FileBrowse, este contiene la longitud de la clave especificada en el método startBrowse, y esta longitud se pasa a CICS en posteriores solicitudes de examen para dicho objeto.

No es necesario proporcionar un **REQID** para una operación de examen; cada objeto de examen contiene un REQID exclusivo, que se utiliza automáticamente para todas las solicitudes de examen posteriores correspondientes a ese objeto de examen.

Servicios HTTP y TCP/IP

Los métodos de obtención en clases `HttpHeader`, `NameValueData` y `FormField` devuelven cabeceras HTTP, pares de nombre y valor y valores de campo de formulario para los mandatos de la API adecuados.

Métodos	Clase de JCICS	Mandatos de `EXEC CICS`
get*()	CertificateInfo	EXTRACT CERTIFICATE / EXTRACT TCPIP
get*()	HttpRequest	EXTRACT WEB
getHeader()	HttpRequest	WEB READ HTTPHEADER
getFormField()	HttpRequest	WEB READ FORMFIELD
getContent()	HttpRequest	WEB RECEIVE
getQueryParm()	HttpRequest	WEB READ QUERYPARM
startBrowseHeader()	HttpRequest	WEB STARTBROWSE HTTPHEADER
getNextHeader()	HttpRequest	WEB READNEXT HTTPHEADER
endBrowseHeader()	HttpRequest	WEB ENDBROWSE HTTPHEADER
startBrowseFormField()	HttpRequest	WEB STARTBROWSE FORMFIELD
getNextFormField()	HttpRequest	WEB READNEXT FORMFIELD

Métodos	Clase de JCICS	Mandatos de EXEC CICS
endBrowseFormField()	HttpRequest	WEB ENDBROWSE FORMFIELD
startBrowseQueryParm()	HttpRequest	WEB STARTBROWSE QUERYPARM
getNextQueryParm()	HttpRequest	WEB READNEXT QUERYPARM
endBrowseQueryParm()	HttpRequest	WEB ENDBROWSE QUERYPARM
writeHeader()	HttpResponse	WEB WRITE
getDocument()	HttpResponse	WEB RETRIEVE
getCurrentDocument()	HttpResponse	WEB RETRIEVE
sendDocument()	HttpResponse	WEB SEND

Nota: Utilice el método get HttpRequestInstance() para obtener el objeto HttpRequest.

Cada solicitud HTTP entrante procesada por el soporte web de CICS incluye una cabecera HTTP. Si la solicitud utiliza el verbo POST HTTP, también incluye datos de documento. Cada solicitud HTTP generada por el soporte web de CICS incluye una cabecera HTTP y datos de documento.

Para procesar esto, JCICS proporciona los siguientes servicios web y TCP/IP:

Cabecera HTTP
Puede examinar la cabecera HTTP utilizando la clase HttpRequest. Con HTTP en modalidad GET, si un cliente ha rellenado un formulario HTTP y ha seleccionado el botón de someter, se envía la serie de consulta.

SSL El soporte web de CICS proporciona la clase TcpipRequest, que se amplía mediante HttpRequest para obtener más información sobre qué cliente sometió la solicitud, así como información básica sobre el soporte SSL. Si se proporciona un certificado SSL, puede utilizar la clase CertificateInfo para examinarlo con detalle.

Documentos
Si se publica un documento en el servidor (HTTP POST), este se proporciona como un documento de CICS. Puede acceder al mismo llamando al método getDocument() en la clase HttpRequest. Consulte “Servicios de documento” en la página 63 para obtener más información sobre el proceso de documentos existentes.

Para proporcionar contenido web del cliente HTTP resultante de una solicitud, el programador de un servidor tiene que crear un documento de CICS utilizando la API de servicios de documento y llamar al método sendDocument().

Para obtener más información sobre el soporte web de CICS, consulte Descripción general de Internet en la Guía de Internet. Para obtener más información sobre las clases web de JCICS, consulte la *Referencia de clases JCICS*.

Servicios de programa

JCICS soporta los mandatos de control de programa de CICS LINK, RETURN, XCTL y SUSPEND.

Para obtener información sobre herramientas que permiten a los programas Java acceder a datos de aplicación de CICS existentes, consulte Interacción con datos estructurados de Java en Aplicaciones Java en CICS.

Tabla 4 lista los métodos y las clases de JCICS que se correlacionan con los mandatos de control de programa de CICS.

Tabla 4. Relación entre métodos, clases de JCICS y mandatos de CICS

Métodos	Clase de JCICS	Mandatos de EXEC CICS
link()	Program	LINK
setNextTransaction(), setNextCOMMAREA(), setNextChannel()	TerminalPrincipalFacility	RETURN
xctl()	Program	XCTL
	No está soportado.	SUSPEND

LINK y XCTL
Puede transferir el control a otro programa que esté definido para CICS utilizando los métodos link() y xctl(). El programa destino puede estar en cualquier lenguaje soportado por CICS.

Si utiliza el método xctl(), se genera una TransferOfControlException para el programa emisor, incluso si se completa correctamente.

RETURN
Solo se soportan los aspectos pseudoconversacionales de este mandato. No es necesario realizar una llamada de CICS para volver; la aplicación puede terminar de la manera normal. Las funciones pseudoconversacionales se soportan mediante métodos de la clase TerminalPrincipalFacility: setNextTransaction() es equivalente a utilizar la opción TRANSID de RETURN, setNextCOMMAREA() es equivalente a utilizar la opción COMMAREA, mientras que setNextChannel() es equivalente a utilizar la opción CHANNEL. Estos métodos se pueden invocar en cualquier momento durante la ejecución del programa y son efectivos cuando el programa termina.

Nota: La longitud de la COMMAREA se utiliza como el valor de LENGTH para CICS. Este valor no debe superar los 32.500 bytes si la COMMAREA se va a pasar entre dos servidores cualquiera de CICS (de cualquier combinación de producto/versión/release). Este límite permite la COMMAREA de 32.500 bytes y deja espacio para cabeceras.

Servicios de planificación

JCICS proporciona soporte para los servicios de planificación de CICS, que le permiten recuperar datos almacenados para una tarea, cancelar solicitudes de control de intervalo e iniciar una tarea a una hora especificada.

Métodos	Clase de JCICS	Mandatos de EXEC CICS
cancel()	StartRequest	CANCEL
retrieve()	Task	RETRIEVE
issue()	StartRequest	START

Para definir qué va a recuperar el método Task.retrieve(), utilice un objeto java.util.BitSet. La clase com.ibm.cics.server.RetrieveBits define los bits que se pueden definir en el objeto BitSet; estos son:
- RetrieveBits.DATA
- RetrieveBits.RTRANSID
- RetrieveBits.RTERMID
- RetrieveBits.QUEUE

Estas se corresponden con las opciones del mandato RETRIEVE de **EXEC CICS**.

El método Task.retrieve() recupera hasta cuatro fragmentos de información en una única invocación, en función de los valores de RetrieveBits. Los datos de DATA, RTRANSID, RTERMID y QUEUE se colocan en un objeto RetrievedData, que se mantiene en un objeto RetrievedDataHolder. El siguiente ejemplo recupera los datos y el ID de transacción:

```
BitSet bs = new BitSet();
bs.set(RetrieveBits.DATA, true);
bs.set(RetrieveBits.RTRANSID, true);
RetrievedDataHolder rdh = new RetrievedDataHolder();
t.retrieve(bs, rdh);
byte[] inData = rdh.value.data;
String transid = rdh.value.transId;
```

Servicios de serialización

JCICS brinda soporte para los servicios de serialización de CICS, que permiten planificar la utilización de un recurso por una tarea.

Métodos	Clase de JCICS	Mandatos de EXEC CICS
dequeue()	SynchronizationResource	DEQ
enqueue(), tryEnqueue()	SynchronizationResource	ENQ

Servicios de almacenamiento

No se brinda ningún soporte para la gestión explícita del almacenamiento mediante servicios de CICS (como **EXEC CICS** GETMAIN). Descubrirá que los recursos estándar de gestión de almacenamiento de Java son suficientes para satisfacer las necesidades para almacenamiento privado de tareas.

La compartición de datos entre tareas se debe realizar utilizando recursos de CICS.

Generalmente los nombres se representan como series Java o matrices de bytes; debe asegurarse de que tienen la longitud necesaria.

Servicios de cola de almacenamiento temporal

JCICS soporta los mandatos de almacenamiento temporal de CICS: DELETEQ TS, READQ TS y WRITEQ TS.

Interacción entre métodos de JCICS y mandatos de EXEC CICS

Para obtener información sobre herramientas que permiten a los programas Java acceder a datos de aplicación de CICS existentes, consulte Interacción con datos estructurados de Java en Aplicaciones Java en CICS.

Tabla 5 lista los métodos y las clases de JCICS que se correlacionan con los mandatos de almacenamiento temporal de CICS.

Tabla 5. Relación entre métodos, clases de JCICS y mandatos de CICS

Métodos	Clase de JCICS	Mandatos de EXEC CICS
delete()	TSQ	DELETEQ TS
readItem(), readNextItem()	TSQ	READQ TS
writeItem(), rewriteItem() writeItemConditional() rewriteItemConditional()	TSQ	WRITEQ TS

DELETEQ TS
Puede suprimir una cola de almacenamiento temporal (TSQ) utilizando el método delete() en la clase TSQ.

READQ TS
La opción INTO de CICS no está soportada en los programas Java. Puede leer un elemento específico de una TSQ utilizando los métodos readItem() y readNextItem() en la clase TSQ. Estos métodos toman como uno de sus argumentos una instancia del objeto ItemHolder, que contendrá los datos leídos en una matriz de bytes. El almacenamiento para esta matriz de bytes lo crea CICS y al final del programa se recoge su basura.

WRITEQ TS
Debe proporcionar los datos que grabar en una cola de almacenamiento temporal dentro de una matriz de bytes de Java. Los métodos writeItem() y rewriteItem() se suspenden si se detecta una condición de NOSPACE y esperan hasta que hay espacio disponible para grabar los datos en la cola. Los métodos writeItemConditional() y rewriteItemConditional() no se suspenden en el caso de una condición de NOSPACE, pero devuelven la condición inmediatamente a la aplicación como una NoSpaceException.

Servicios de terminal

JCICS proporciona soporte para estos mandatos de servicios de terminal de CICS.

Métodos	Clase de JCICS	Mandatos de EXEC CICS
converse()	TerminalPrincipalFacility	CONVERSE
	No está soportado.	HANDLE AID
receive()	TerminaPrincipalFacility	RECEIVE
send()	TerminaPrincipalFacility	SEND
	No está soportado.	WAIT TERMINAL

Si una tarea tiene un terminal como recurso principal, CICS crea automáticamente dos PrintWriter Java que se pueden utilizar como secuencias de salida estándar y de error estándar. Están correlacionados con el terminal de la tarea. Las dos secuencias llamadas `out` y `err` son archivos públicos en el objeto Task y se pueden utilizar igual que System.out y System.err.

Los datos que enviar a un terminal deben proporcionarse en una matriz de bytes de Java. Los datos se leen desde el terminal a un objeto DataHolder. CICS proporciona el almacenamiento para los datos devueltos, que se desasignará cuando finalice el programa.

Servicios de cola de datos transitorios

JCICS soporta los mandatos de datos transitorios de CICS, DELETEQ TD, READQ TD y WRITEQ TD. Todas las opciones están soportadas excepto la opción INTO.

Interacción entre métodos de JCICS y mandatos de EXEC CICS

Para obtener información sobre herramientas que permiten a los programas Java acceder a datos de aplicación de CICS existentes, consulte Interacción con datos estructurados de Java en Aplicaciones Java en CICS.

La Tabla 6 lista los métodos y las clases de JCICS que se correlacionan con los mandatos de datos transitorios de CICS.

Tabla 6. Relación entre métodos, clases de JCICS y mandatos de CICS

Métodos	Clase de JCICS	Mandatos de EXEC CICS
delete()	TDQ	DELETEQ TD
readData(), readDataConditional()	TDQ	READQ TD
writeData()	TDQ	WRITEQ TD

DELETEQ TD
Puede suprimir una cola de datos transitoria (TDQ) utilizando el método delete() en la clase TDQ.

READQ TD
La opción INTO de CICS no está soportada en los programas Java. Puede leer desde una TDQ utilizando el método readData() o readDataConditional() de la clase TDQ. Estos métodos toman como parámetro una instancia del objeto DataHolder que contendrá los datos leídos en una matriz de bytes. El almacenamiento para esta matriz de bytes lo crea CICS y al final del programa se recoge su basura.

El método readDataConditional() controla la lógica NOSUSPEND de CICS. Si se detecta una condición de QBUSY, se devuelve a la aplicación inmediatamente como una QueueBusyException.

El método readData() se suspende si intenta acceder a un registro que esté en uso por otra tarea y no hay más registros confirmados.

WRITEQ TD
Debe proporcionar los datos que grabar en una TDQ dentro de una matriz de bytes de Java.

Servicios de unidad de trabajo (UOW)

JCICS proporciona soporte para el servicio SYNCPOINT de CICS.

Tabla 7. Relación entre JCICS y mandatos de EXEC CICS para servicios de UOW

Métodos	Clase de JCICS	Mandatos de EXEC CICS
commit(), rollback()	Task	SYNCPOINT

Servicios web

JCICS soporta todos los mandatos de la API que están disponibles para trabajar con los servicios web de una aplicación.

Métodos	Clase de JCICS	Mandatos EXEC CICS
invoke()	WebService	INVOKE WEBSERVICE
create()	SoapFault	SOAPFAULT CREATE
addFaultString()	SoapFault	SOAPFAULT ADD FAULTSTRING
addSubCode()	SoapFault	SOAPFAULT ADD SUBCODESTR
delete()	SoapFault	SOAPFAULT DELETE
create()	WSAEpr	WSAEPR CREATE
delete()	WSAContext	WSACONTEXT DELETE

Métodos	Clase de JCICS	Mandatos EXEC CICS
set*()	WSAContext	WSACONTEXT BUILD
get*()	WSAContext	WSACONTEXT GET

El siguiente ejemplo muestras cómo puede utilizar JCICS para crear una solicitud de servicio web:

```
Channel requesterChannel = Task.getTask().createChannel("TestRequester");
                    Container appData = requesterChannel.createContainer("DFHWS-DATA");
                    byte[] exampleData = "ExampleData".getBytes();
                    appData.put(exampleData);

                    WebService requester = new WebService();
                    requester.setName("MyWebservice");
                    requester.invoke(requesterChannel, "myOperationName");

                    byte[] response = appData.get();
```

Para gestionar los datos de aplicación que se envían y se reciben en una solicitud de servicio web, puede utilizar una herramienta como JZOS para generar clases si está trabajando con datos estructurados. Para obtener más información, consulte "Interacción con datos estructurados procedentes de Java" en la página 51. También puede utilizar Java para generar y consumir XML directamente.

Correlación de excepciones de JCICS

En Java, una condición devuelta por un mandato de CICS se correlaciona con una excepción de Java.

Tabla 8. Correlación de excepciones de Java

Condición de CICS	Excepción de Java	Condición de CICS	Excepción de Java
ALLOCERR	AllocationErrorException	CBIDERR	InvalidControlBlockIdException
CCSIDERR	CCSIDErrorException	CHANNELERR	ChannelErrorException
CONTAINERERR	ContainerErrorException	DISABLED	FileDisabledException
DSIDERR	FileNotFoundException	DSSTAT	DestinationStatusChangeException
DUPKEY	DuplicateKeyException	DUPREC	DuplicateRecordException
END	EndException	ENDDATA	EndOfDataException
ENDFILE	EndOfFileException	ENDINPT	EndOfInputIndicatorException
ENQBUSY	ResourceUnavailableException	ENVDEFERR	InvalidRetrieveOptionException
EOC	EndOfChainIndicatorException	EODS	EndOfDataSetIndicatorException
EOF	EndOfFileIndicatorException	ERROR	ErrorException
EXPIRED	TimeExpiredException	FILENOTFOUND	FileNotFoundException
FUNCERR	FunctionErrorException	IGREQID	InvalidREQIDPrefixException
IGREQCD	InvalidDirectionException	ILLOGIC	LogicException
INBFMH	InboundFMHException	INVERRTERM	InvalidErrorTerminalException
INVEXITREQ	InvalidExitRequestException	INVLDC	InvalidLDCException
INVMPSZ	InvalidMapSizeException	INVPARTNSET	InvalidPartitionSetException
INVPARTN	InvalidPartitionException	INVREQ	InvalidRequestException
INVTSREQ	InvalidTSRequestException	IOERR	IOErrorException
ISCINVREQ	ISCInvalidRequestException	ITEMERR	ItemErrorException

Tabla 8. Correlación de excepciones de Java (continuación)

Condición de CICS	Excepción de Java	Condición de CICS	Excepción de Java
JIDERR	InvalidJournalIdException	LENGERR	LengthErrorException
MAPERROR	MapErrorException	MAPFAIL	MapFailureException
NAMEERROR	NameErrorException	NODEIDERR	InvalidNodeIdException
NOJBUFSP	NoJournalBufferSpaceException	NONVAL	NotValidException
NOPASSBKRD	NoPassbookReadException	NOPASSBKWR	NoPassbookWriteException
NOSPACE	NoSpaceException	NOSPOOL	NoSpoolException
NOSTART	StartFailedException	NOSTG	NoStorageException
NOTALLOC	NotAllocatedException	NOTAUTH	NotAuthorisedException
NOTFND	RecordNotFoundException	NOTOPEN	NotOpenException
OPENERR	DumpOpenErrorException	OVERFLOW	MapPageOverflowException
PARTNFAIL	PartitionFailureException	PGMIDERR	InvalidProgramIdException
QBUSY	QueueBusyException	QIDERR	InvalidQueueIdException
QZERO	QueueZeroException	RDATT	ReadAttentionException
RETPAGE	ReturnedPageException	ROLLEDBACK	RolledBackException
RTEFAIL	RouteFailedException	RTESOME	RoutePartiallyFailedException
SELNERR	DestinationSelectionErrorException	SESSBUSY	SessionBusyException
SESSIONERR	SessionErrorException	SIGNAL	InboundSignalException
SPOLBUSY	SpoolBusyException	SPOLERR	SpoolErrorException
STRELERR	STRELERRException	SUPPRESSED	SuppressedException
SYMBOLERR	SymbolErrorException	SYSBUSY	SystemBusyException
SYSIDERR	InvalidSystemIdException	TASKIDERR	InvalidTaskIdException
TCIDERR	TCIDERRException	TEMPLATERR	TemplateErrorException
TERMERR	TerminalException	TERMIDERR	InvalidTerminalIdException
TOKENERR	TokenErrorException		
TRANSIDERR	InvalidTransactionIdException	TSIOERR	TSIOErrorException
UNEXPIN	UnexpectedInformationException	USERIDERR	InvalidUserIdException
WRBRK	WriteBreakException	WRONGSTAT	WrongStatusException

Nota: `NonHttpDataException` la genera `getContent()` si el mandato WEB RECEIVE de CICS indica que los datos recibidos están en un mensaje que no es HTTP (definiendo TYPE=HTTPNO).

Utilización de JCICS

Las clases de la biblioteca JCICS se utilizan como las clases Java normales. Sus aplicaciones declaran una referencia del tipo necesario y se crea una nueva instancia de una clase utilizando el operador `new`.

Acerca de esta tarea

Los recursos de CICS reciben el nombre mediante el método setName para proporcionar el nombre del recurso subyacente de CICS. Tras crear el recurso, puede manipular objetos utilizando construcciones estándar de Java. Puede llamar

a métodos de los objetos declarados del modo normal. Hay disponibles detalles completos de los métodos soportados para cada clase en el Javadoc proporcionado.

No utilice finalizadores en programas Java de CICS. Para obtener una explicación de por qué no se recomiendan finalizadores, consulte la Java Guía de diagnósticos.

No finalice programas Java de CICS emitiendo una llamada a System.exit(). Cuando las aplicaciones Java se ejecutan en CICS, se llama al método public static void main() mediante el uso de otro programa Java llamado derivador Java. Cuando se utiliza el derivador, CICS inicializa el entorno para aplicaciones Java y, lo que es más importante, limpia cualquier proceso que se utilice durante la vida de la aplicación. La terminación de la JVM, incluso con un código de retorno de limpieza de 0, impide que se ejecute este proceso de limpieza, lo que puede dar origen a inconsistencia de datos. Utilizar System.exit() cuando la aplicación se está ejecutando en un servidor de JVM termina el servidor de JVM y desactiva temporalmente CICS de inmediato.

Procedimiento

1. Escriba el método principal. CICS intenta pasar el control al método con una firma main(CommAreaHolder) en la clase especificada por el atributo JVMCLASS del recurso PROGRAM. Si no se encuentra este método, CICS intenta invocar el método main(String[]).
2. Para crear objetos utilizando JCICS, siga estos pasos:
   a. Declare una referencia:
      ```
      TSQ tsq;
      ```
   b. Utilice el operador new para crear un objeto:
      ```
      tsq = new TSQ()
      ```
   c. Utilice el método setName para dar un nombre al objeto:
      ```
      tsq.setName("JCICSTSQ");
      ```
3. Utilice el objeto para interaccionar con CICS.

Ejemplo

Este ejemplo muestra cómo crear un objeto TSQ, invocar el método delete en el objeto de cola de almacenamiento temporal que acaba de crear y detectar la excepción generada si la cola está vacía.

```
// Defina un nombre de paquete para el paquete
de programa unit_test;

// Importe el paquete de JCICS.
import com.ibm.cics.server.*;

// Declare una clase para una aplicación CICS.
public class JCICSTSQ
{

    // Se llama al método principal cuando se ejecuta la aplicación.
    public static void main(CommAreaHolder cah)
    {

         try
         {
             // Cree un objeto de cola de almacenamiento temporal y proporciónele
             un nombre.
             TSQ tsq = new TSQ();
             tsq.setName("JCICSTSQ");

             // Suprima la cola, si existe.
```

```
                try
                {
                      tsq.delete();
                }
                catch(InvalidQueueIdException e)
                {
                     // Absorba QIDERR.
                     System.out.println("QIDERR ignored!");
                }

                // Escriba un elemento para la cola.
                String transaction = Task.getTask().getTransactionName();
                String message = "Transaction name is - " + transaction;
                tsq.writeItem(message.getBytes());

            }
            catch(Throwable t)
            {
                System.out.println("Unexpected Throwable: " + t.toString());
            }

            // Devolución de la aplicación.
            return;
        }
}
```

Restricciones de Java

Cuando esté desarrollando aplicaciones Java, debe respetar ciertas restricciones para evitar problemas cuando la aplicación se está ejecutando en CICS.

Las aplicaciones Java que se ejecutan en CICS están sujetos a las siguientes restricciones:

- No puede utilizar el método System.exit() en sus aplicaciones Java. Si utiliza este método, la aplicación finaliza de forma anómala. El servidor de JVM y CICS también se apagan.
- No puede utilizar llamadas de API JCICS en las clases activator de los paquetes OSGi.
- Los métodos start y stop en activadores de paquete deben volver en una cantidad de tiempo razonable.

Acceso a datos desde aplicaciones Java

Puede escribir aplicaciones Java que sean capaces de acceder a datos en DB2 y VSAM y actualizarlos. Como alternativa, puede enlazar con programas en otros lenguajes para acceder a DB2, VSAM e IMS.

Puede utilizar cualquiera de las siguientes técnicas al escribir una aplicación Java para acceder a datos en CICS. El gestor de recuperación de CICS mantiene la integridad de los datos.

Acceso a datos relacionales

Puede escribir una aplicación Java para acceder a datos relacionales en DB2 mediante cualquiera de los métodos siguientes:

- Un mandato **JCICS LINK** o el CCI Connector for CICS TS, para enlazar con un programa que utilice mandatos de lenguaje de consulta estructurado (SQL) para acceder a los datos.

- Si hay disponible un controlador adecuado, utilice llamadas a Java Data Base Connectivity (JDBC) o Structured Query Language for Java (SQLJ) para acceder a los datos directamente. Hay controladores JBDC adecuados disponibles para DB2. Para obtener más información sobre la utilización de interfaces de programación de aplicaciones JBDC y SQLJ, consulte Uso de JDBC y SQLJ para acceder a datos de DB2 desde programas Java en la Guía DB2.
- JavaBeans que utilicen JDBC o SQLJ como el mecanismo de acceso subyacente. Puede utilizar cualquier entorno de desarrollo integrado (IDE) de Java adecuado para desarrollar este tipo de JavaBeans.
- Beans de entidad. CICS no tiene soporte para beans de entidad que se ejecuten en CICS, pero sí que lo tiene para el acceso a beans de entidad que se ejecuten en otros servidores EJB. Un enterprise bean de CICS podría, por ejemplo, utilizar un bean de entidad que se ejecute en WebSphere Application Server para acceder a DB2 en z/OS.

Acceso a datos de Data Language/I

Para acceder a datos de DL/I en IMS, la aplicación Java debe utilizar un mandato **JCICS LINK** para enlazar con un programa intermedio que emita mandatos EXEC DLI para acceder a los datos.

Acceso a datos de VSAM

Para acceder a datos de VSAM, una aplicación Java puede utilizar cualquiera de los métodos siguientes:

- Las clases de control de archivos de JCICS para accede a VSAM directamente.
- Un mandato **JCICS LINK** o el CCI Connector for CICS TS, para enlazar con un programa que emita mandatos de control de archivos de CICS para acceder a los datos.

Conectividad desde aplicaciones Java en CICS

Los programas Java en el entorno CICS pueden abrir sockets TCP/IP y comunicarse con procesos externos. Puede utilizar programas Java como una pasarela para conectar con otras aplicaciones empresariales que tal vez no estén disponibles para los programas de CICS en otros lenguajes. Por ejemplo, puede escribir un programa Java para comunicarse con una base de datos o un servlet remotos.

En algunos casos, esta conectividad se integra con CICS para brindar calidades de servicio empresariales, como transacciones distribuidas y propagación de identidad. En otros casos, puede utilizar la conectividad sin transacciones distribuidas y otros servicios que proporciona CICS. En función del tipo de conectividad que necesite, puede haber disponibles productos de otros proveedores que permitan la conectividad con aplicaciones empresariales que CICS no soporte de forma nativa.

En general, las JVM en un entorno de CICS tienen unas prestaciones similares a las de las JVM en modalidad de proceso por lotes. Una JVM en modalidad de proceso por lotes se ejecuta como un proceso autónomo fuera del entorno de CICS y, normalmente, se lanza desde una línea de mandatos de UNIX System Services o mediante un trabajo de JCL. La mayoría de las aplicaciones que pueden trabajar en una JVM en modalidad de proceso por lotes también pueden ejecutarse en una JVM en CICS en la misma medida. Por ejemplo, si escribe una aplicación Java en modalidad de proceso por lotes para comunicarse con una base de datos que no

sea de IBM mediante un controlador JDBC de terceros, es probable que la misma aplicación funcione en una JVM en CICS. Si desea utilizar el código que proporciona el proveedor como controladores JBDC no de IBM en una JVM en CICS, consulte con su proveedor para determinar si cuentan con soporte para que su código se ejecute en una JVM en CICS.

Algunas aplicaciones en modalidad de proceso por lotes pueden comportarse de modo distinto cuando están alojadas en una JVM en CICS, debido al modo en el que CICS reutiliza las JVM. Cualquier dato almacenado en variables estáticas persiste en los distintos usos de la JVM. Para obtener más información acerca del comportamiento de aplicaciones Java en CICS, consulte “Entorno de tiempo de ejecución Java en CICS” en la página 33.

Las aplicaciones en modalidad de proceso por lotes que se ejecutan en una JVM en el entorno CICS generalmente no aprovechan las prestaciones de CICS. Por ejemplo, si un programa Java en CICS actualiza registros en una base de datos que no sea de IBM mediante un controlador JDBC de terceros, CICS no sabe que se produce esta actividad y, por tanto, no intenta incluir las actualizaciones en la transacción actual de CICS.

Capítulo 4. Configuración del soporte de Java

Realice las tareas de configuración básicas para el soporte de Java en la región CICS.

Antes de empezar

Los componentes de Java que son necesarios para CICS se configuran durante la instalación del producto. Debe asegurarse de que los componentes de Java se instalan correctamente utilizando la información que aparece en Comprobación de la instalación de sus componentes Java en la Guía de instalación.

Procedimiento

1. Defina el parámetro de inicialización del sistema JVMPROFILEDIR a un directorio adecuado de z/OS UNIX en el que desea almacenar los perfiles de JVM utilizados por la región CICS. Para obtener más detalles, consulte "Definición de la ubicación de los perfiles de JVM".
2. Asegúrese de que la región CICS tiene suficiente memoria para ejecutar aplicaciones Java. Para obtener más detalles, consulte "Definición de los límites de memoria para Java" en la página 82.
3. Conceda a su región CICS permiso para acceder a los recursos mantenidos en z/OS UNIX, incluidos los perfiles de JVM, los directorios y los archivos necesarios para crear una JVM. Para obtener más detalles, consulte "Concesión a las regiones CICS de acceso a directorios y archivos z/OS UNIX" en la página 83.

Resultados

Ha configurado la región CICS para que soporte Java.

Qué hacer a continuación

Si está actualizando aplicaciones Java existentes, siga las pautas que se muestran en Actualización. Para crear un servidor de JVM o JVM en agrupación para ejecutar cargas de trabajo de Java, consulte Capítulo 5, "Habilitación de aplicaciones para que utilicen una JVM", en la página 87.

Definición de la ubicación de los perfiles de JVM

CICS carga los perfiles de JVM desde el directorio de z/OS UNIX que se especifica mediante el parámetro de inicialización del sistema **JVMPROFILEDIR**. Debe modificar el valor del parámetro **JVMPROFILEDIR** a una nueva ubicación y copiar los perfiles de JVM de ejemplo provistos a este directorio, de forma que pueda utilizarlos para verificar su instalación.

Antes de empezar

El parámetro de inicialización del sistema USSHOME debe especificar el directorio raíz de los archivos CICS en z/OS UNIX.

Acerca de esta tarea

Los perfiles de JVM de ejemplo que CICS proporciona se personalizan para su sistema durante el proceso de instalación de CICS, de forma que pueda utilizarlos inmediatamente para verificar la instalación. Puede personalizar copias de estos archivos para sus propias aplicaciones Java.

Los valores que resultan adecuados en perfiles de JVM pueden cambiar de una versión de CICS a otra, por lo que para facilitar la determinación de problemas, utilice los ejemplos proporcionados por CICS como base para todos los perfiles. Compruebe la información de actualización para descubrir qué opciones son nuevas o se han modificado en los perfiles de JVM.

Procedimiento

1. Defina el parámetro de inicialización del sistema JVMPROFILEDIR en la ubicación de z/OS UNIX en la que desea almacenar los perfiles de JVM utilizados por la región CICS. El valor que se especifique puede tener hasta 240 caracteres de largo.

   El valor proporcionado para el parámetro de inicialización del sistema **JVMPROFILEDIR** es `/usr/lpp/cicsts/cicsts42/JVMProfiles`, que es la ubicación de instalación para los perfiles de JVM de ejemplo. Este directorio no es un lugar seguro para almacenar los perfiles de JVM personalizados, ya que se arriesga a perder los cambios si los perfiles de JVM de ejemplo se sobrescriben cuando se aplica el mantenimiento del programa. Por ello, siempre debe cambiar **JVMPROFILEDIR** para que especifique un directorio de z/OS UNIX distinto en el que pueda almacenar sus perfiles de JVM. Elija un directorio en el que pueda otorgar los permisos adecuados a los usuarios que deban personalizar los perfiles de JVM.
2. Copie los perfiles de JVM de ejemplo proporcionados por CICS, DFHJVMPR, DFHJVMAX, DFHOSGI, y DFHJVMCD, desde su ubicación de instalación al directorio de z/OS UNIX. El perfil DFHJVMCD, aunque estrictamente no es un perfil de JVM de ejemplo, es necesario para transacciones internas de CICS Java y para gestionar la memoria caché de clase compartida.

   Cuando se instala CICS, los perfiles de JVM de ejemplo proporcionados por CICS se colocan en el directorio `/usr/lpp/cicsts/cicsts42/JVMProfiles`. El directorio `/usr/lpp/cicsts/cicsts42` es el directorio de instalación de los archivos de CICS en z/OS UNIX. Este directorio se especifica mediante el parámetro **USSDIR** en el trabajo de instalación DFHISTAR.

Definición de los límites de memoria para Java

Las aplicaciones Java necesitan más memoria que los programas escritos en otros lenguajes. Debe asegurarse de que CICS y Java tienen almacenamiento y memoria suficientes disponibles para ejecutar aplicaciones Java.

Acerca de esta tarea

Java utiliza almacenamiento por debajo de la línea de los 16 MB, almacenamiento de 31 bits y almacenamiento de 64 bits. El almacenamiento necesario para el almacenamiento dinámico de la JVM procede del almacenamiento de la región de CICS de MVS y no de EDSA.

Procedimiento

1. Asegúrese de que el parámetro **MEMLIMIT** de z/OS está definido a un valor adecuado. Este parámetro limita la cantidad de almacenamiento de 64 bits que

puede utilizar el espacio de direcciones de CICS. CICS utiliza la versión de 64 bits de Java y debe asegurarse de que **MEMLIMIT** se define en un valor suficientemente grande para este y otros recursos de CICS que utilicen almacenamiento de 64 bits.

Consulte los siguientes temas:

- "Cálculo de los requisitos de almacenamiento para JVM en agrupación" en la página 177
- "Cálculo de los requisitos de almacenamiento para servidores de JVM" en la página 168
- Estimación, comprobación y configuración de MEMLIMIT en la Guía de rendimiento

2. Asegúrese de que el parámetro **REGION** de la secuencia de trabajos de inicio es lo suficientemente grande para ejecutar Java. Cada JVM necesita almacenamiento por debajo de la línea de 16 MB para ejecutar aplicaciones, incluyendo código compilado just-in-time, y almacenamiento de trabajo para pasar parámetros a CICS.

Concesión a las regiones CICS de acceso a directorios y archivos z/OS UNIX

CICS necesita tener acceso a directorios y archivos z/OS UNIX. Durante la instalación, a cada una de las regiones CICS se le asigna un identificador de usuario (UID) de z/OS UNIX. Las regiones se conectan a un grupo RACF al que se le asigna un identificador de grupo (GID) de z/OS UNIX. Utilice el UID y el GID para otorgar permiso a la región CICS para que acceda a los directorios y los archivos z/OS UNIX.

Antes de empezar

Asegúrese de ser un superusuario de z/OS UNIX o el propietario de los directorios y archivos. El propietario de los directorios y archivos inicialmente se define como el UID del programador del sistema que instala el producto. El propietario de los directorios y archivos debe estar conectado al grupo RACF al que se asignó un GID durante la instalación. El propietario puede tener ese grupo RACF como su grupo predeterminado (DLFTGRP) o puede estar conectado a él como uno de sus grupos complementarios.

Acerca de esta tarea

z/OS UNIX System Services trata cada región CICS como un usuario UNIX. Puede otorgar a los usuarios permisos para acceder a directorios y archivos z/OS UNIX de distintos modos. Por ejemplo, puede conceder los permisos de grupo adecuados para el directorio o el archivo al grupo RACF al que se conectan las regiones CICS. Esta opción puede ser la mejor para un entorno de producción y se explica en los siguientes pasos.

Procedimiento

1. Identifique los directorios y archivos z/OS UNIX a los que tienen que acceder sus regiones CICS.

Directorios predeterminados	Permiso	Descripción
`/usr/lpp/java/J6.0.1_64/bin`	lectura y ejecución	Directorios de IBM 64 bits SDK para z/OS, Java Technology Edition

Directorios predeterminados	Permiso	Descripción
/usr/lpp/java/J6.0.1_64/bin/j9vm	lectura y ejecución	Directorios de IBM 64 bits SDK para z/OS, Java Technology Edition
/usr/lpp/cicsts/cicsts42	lectura y ejecución	El directorio de instalación para los archivos CICS en z/OS UNIX. Los archivos de este directorio incluyen perfiles de ejemplo y archivos JAR proporcionados por CICS.
/u/*ID de usuario de región CICS*	lectura, grabación y ejecución	El directorio de trabajo de la región CICS. Este directorio contiene entrada, salida y mensajes de las JVM.
/usr/lpp/cicsts/cicsts42/JVMProfiles/	lectura y ejecución	Directorio que contiene los perfiles de JVM para la región CICS, como se especifica en el parámetro de inicialización del sistema **JVMPROFILEDIR**.

2. Liste los directorios y archivos para mostrar los permisos. Vaya al directorio en el que quiere empezar y emita el siguiente mandato de UNIX:

   ```
   ls -la
   ```

   Si se emite este mandato en el entorno de shell de z/OS UNIX System Services cuando el directorio actual es el directorio de inicio de CICSHT##, puede ver una lista como la del siguiente ejemplo:

   ```
   /u/cicsht##:>ls -la
   total 256
   drwxr-xr-x   2 CICSHT## CICSTS42     8192 Mar 15  2008 .
   drwx------   4 CICSHT## CICSTS42     8192 Jul  4 16:14 ..
   -rw-------   1 CICSHT## CICSTS42     2976 Dec  5  2010 Snap0001.trc
   -rw-r--r--   1 CICSHT## CICSTS42     1626 Jul 16 11:15 dfhjvmerr
   -rw-r--r--   1 CICSHT## CICSTS42        0 Mar 15  2010 dfhjvmin
   -rw-r--r--   1 CICSHT## CICSTS42      458 Oct  9 14:28 dfhjvmout
   /u/cicsht##:>
   ```

3. Si utiliza los permisos de grupo para conceder acceso, compruebe que los permisos de grupo correspondientes a cada directorio y archivo conceden el nivel de acceso que CICS necesita para el recurso. Los permisos se indican, en tres conjuntos, mediante los caracteres `r`, `w`, `x` y `-`. Estos caracteres representan lectura, grabación, ejecución y ninguno, y se muestran en la columna izquierda de la línea de mandatos, empezando por el segundo carácter. El primer conjunto son los permisos de propietario, el segundo conjunto son los permisos de grupo y el tercer conjunto son otros permisos. En el ejemplo anterior, el propietario tiene permisos de lectura y grabación para `dfhjvmerr`, `dfhjvmin` y `dfhjvmout`, pero el grupo y todos los demás solo tienen permisos de lectura.
4. Si desea cambiar los permisos de grupo correspondientes a un recurso, utilice el mandato de UNIX `chmod`. El siguiente ejemplo define los permisos de grupo para el directorio indicado y sus subdirectorios para lectura, grabación y ejecución. `-R` aplica permisos de forma recursiva a todos los subdirectorios y los archivos:

   ```
   chmod -R g=rwx directorio
   ```

   El siguiente ejemplo define los permisos de grupo para el archivo indicado para lectura y ejecución:

   ```
   chmod g+rx nombre_archivo
   ```

El siguiente ejemplo desactiva el permiso de grabación para el grupo en dos archivos indicados:

```
chmod g-w nombre_archivo nombre_archivo
```

En todos estos ejemplos, `g` designa permisos de grupo. Si desea corregir otros permisos, `u` designa permisos de usuario (propietario) y `o` designa otros permisos.

5. Asigne los permisos de grupo para cada recurso al grupo RACF que elija para que sus regiones CICS accedan a UNIX. Es necesario asignar permisos de grupo para cada directorio y sus subdirectorios, así como para los archivos que hay en ellos. Especifique el siguiente mandato de UNIX:

   ```
   chgrp -R GID directorio
   ```

   GID es el GID numérico del grupo de RACF y *directorio* es la vía de acceso completa de un directorio al que desea asignar los permisos de las regiones CICS. Por ejemplo, para asignar los permisos de grupo para el directorio `/usr/lpp/cicsts/cicsts42`, utilice el siguiente mandato:

   ```
   chgrp -R GID /usr/lpp/cicsts/cicsts42
   ```

   Como los ID de usuario de su región CICS están conectados con el grupo RACF, las regiones CICS tienen los permisos adecuados para todos esos directorios y archivos.

Resultados

Se ha asegurado de que CICS cuenta con los permisos adecuados para acceder a los directorios y los archivos z/OS UNIX para ejecutar aplicaciones Java.

Cuando se cambie el recurso de CICS que se está configurando, como al mover archivos o crear archivos nuevos, recuerde repetir este procedimiento para asegurarse de que sus regiones CICS cuentan con permiso para acceder a los archivos nuevos o movidos.

Qué hacer a continuación

Verifique que el soporte de Java está configurado correctamente mediante los programas y los perfiles de ejemplo.

Capítulo 5. Habilitación de aplicaciones para que utilicen una JVM

Al igual que para las aplicaciones que no son Java, CICS requiere que se definan los recursos necesarios para ejecutar un programa Java en una JVM. También se debe definir dónde encontrar las clases para la aplicación.

Puede ejecutar aplicaciones Java estándar en una JVM en agrupación o en un servidor de JVM creando un recurso PROGRAM. Utilice una JVM en agrupación solo si la aplicación no es de enhebramiento seguro. En caso contrario, utilice un servidor de JVM para aplicaciones Java.

Los objetos sin estado CORBA y los enterprise beans no cuentan con sus propios recursos PROGRAM, sino que utilizan el perfil especificado por el programa procesador de solicitudes. Los objetos sin estado CORBA y los enterprise beans se ejecutan únicamente en una JVM en agrupación.

Configuración de un servidor de JVM

Para ejecutar aplicaciones Java o Axis2 en un servidor de JVM, debe configurar los recursos de CICS y crear un perfil de JVM que transfiera opciones a la JVM.

Acerca de esta tarea

Un servidor de JVM puede gestionar varias solicitudes simultáneas para distintas aplicaciones Java en una única JVM. El recurso JVMSERVER representa al servidor de JVM en CICS. El recurso define el perfil de JVM que especifica opciones para la JVM, el programa que proporciona valores para el enclave de Language Environment y el límite de hebras. Un servidor de JVM puede ejecutar distintos tipos de carga de trabajo:

- Puede configurar el servidor de JVM para ejecutar aplicaciones que están empaquetadas como paquetes OSGi.
- Puede configurar el servidor de JVM para ejecutar el proceso SOAP de los servicios web con el motor Axis2 SOAP.

Un servidor de JVM puede ejecutar ambos tipos de carga de trabajo; por lo tanto, se proporciona un perfil de JVM para cada tipo de carga de trabajo. Cualquier cambio que se realice en estos perfiles se aplicará a todos los servidores de JVM que lo utilicen. El perfil de JVM DFHOSGI contiene las opciones para ejecutar una infraestructura OSGi en el servidor de JVM. Cuando personalice el perfil DFHOSGI, asegúrese de que los cambios son adecuados para todas las aplicaciones Java que utilicen el servidor de JVM. El perfil de JVM DFHJVMAX contiene las opciones para ejecutar Axis2 en el servidor de JVM.

Procedimiento

1. Cree un perfil de JVM para el servidor de JVM. Puede copiar el perfil suministrado adecuado, DFHJVMAX o DFHOSGI, del directorio de instalación en el directorio que especifique el parámetro de inicialización del sistema **JVMPROFILEDIR**. El perfil que copie no necesita más cambios, pero puede editar las opciones según considere adecuado para su entorno. Si cambia el nombre del perfil, este debe tener entre 1 y 8 caracteres de longitud.

Consejo: Puede utilizar la perspectiva z/OS de CICS Explorer para copiar los perfiles entre directorios.

2. Opcional: Abra el perfil de JVM y edite las opciones si es necesario. En un servidor de JVM solo se soporta un subconjunto de opciones, por lo que debe utilizar la lista de opciones de "Perfiles de JVM: opciones y ejemplos" en la página 104 como guía. Cada parámetro o propiedad se especifica en una línea distinta, y el valor del parámetro o la propiedad se delimita mediante el final de la línea. Siga las reglas de codificación que aparecen en "Reglas para codificación de perfiles de JVM" en la página 106.

   No especifique las opciones de vía de acceso de clases en el perfil DFHOSGI. La infraestructura OSGi determina dónde se ubican las clases para cada aplicación. Puede realizar los siguientes cambios:

   a. Utilice la opción `LIBPATH_SUFFIX` para especificar cualquier directorio que contenga archivos de C nativos de biblioteca de enlace dinámico (DLL) que necesite el servidor de JVM. El middleware y las herramientas que proporcionan IBM u otros proveedores pueden requerir que se añadan archivos DLL a la vía de acceso a biblioteca; por ejemplo, se necesitan archivos DLL para utilizar los controladores JDBC de DB2.
   b. Solo para OSGi, utilice la opción `OSGI_BUNDLES` para especificar paquetes de middleware que desee ejecutar en la infraestructura OSGi. Los paquetes de middleware son un tipo de paquete OSGi que contiene clases para implementar servicios del sistema, como conectar con WebSphere MQ.
   c. Solo para OSGi, utilice la opción `OSGI_FRAMEWORK_TIMEOUT` para especificar cuántos segundos debe esperar CICS a que la infraestructura OSGi se inicialice antes de finalizar el tiempo de espera. El valor predeterminado es 60 segundos. Si la infraestructura necesita más tiempo que el intervalo especificado, el servidor de JVM no se puede inicializar.
   d. Cambie el destino para mensajes, rastreo y salida procedentes de la JVM. Puede cambiar el nombre y la ubicación de los archivos `dfhjvmtrc`, `stdin`, `stdout` y `stderr` y de los volcados de memoria de Java. Para evitar la intercalación de la salida, utilice el símbolo &JVMSERVER; para que estos archivos sean exclusivos de cada servidor de JVM.

3. Guarde los cambios en el perfil de JVM. El perfil de JVM debe guardarse en EBCDIC.
4. Cree un recurso JVMSERVER para el servidor de JVM.
   a. Especifique el nombre del perfil de JVM que ha creado.
   b. Especifique el límite de hebras para el servidor de JVM. El número de hebras necesario depende de la carga de trabajo que desee ejecutar en el servidor de JVM. Para empezar, puede aceptar el valor predeterminado y luego ajustar el entorno. Puede configurar hasta 256 hebras en un servidor de JVM.
   c. Opcional: Especifique el programa que proporciona las opciones de Language Environment para el enclave si es distinto a DFHAXRO. CICS proporciona un conjunto predeterminado de valores que ya está compilado en el programa DFHAXRO. Puede ajustar el enclave proporcionando sus propias opciones si es necesario. Para obtener más información, consulte el apartado "Utilización de DFHAXRO para modificar el enclave de un servidor de JVM" en la página 186.

Resultados

Cuando se habilita el recurso JVMSERVER, CICS crea un enclave de Language Environment y pasa las opciones desde el perfil de JVM al servidor de JVM. En función de las opciones del perfil, el servidor de JVM se configura para ejecutar una infraestructura OSGi o Axis2:

- Si el servidor de JVM admite OSGi, la JVM se lanza y la infraestructura OSGi resuelve los paquetes de middleware de OSGi.
- Si el servidor de JVM admite Axis2, la JVM se lanza y carga los archivos JAR de Axis2.

Cuando el servidor de JVM se inicia de forma correcta, el recurso JVMSERVER se instala con el estado ENABLED.

Si se produce un error (por ejemplo, si CICS no puede encontrar ni leer el perfil de JVM), el servidor de JVM no se puede inicializar. El recurso JVMSERVER se instala en el estado DISABLED y CICS emite mensajes de error para el registro del sistema.

Qué hacer a continuación

Para un servidor de JVM que esté configurado para admitir OSGi, puede instalar paquetes OSGi en la infraestructura, como se describe en "Instalación de paquetes OSGi en un servidor de JVM" en la página 90. Para un servidor de JVM que esté configurado para admitir Axis2, puede configurar CICS para ejecutar solicitudes de servicios web en el servidor de JVM, como se describe en en *Guía de servicios web de CICS*.

Configuración de un servidor de JVM para DB2

Si desea acceder a DB2 desde las aplicaciones de Java que se están ejecutando en un servidor de JVM, debe añadir opciones al perfil de JVM.

Antes de empezar

Para utilizar el servidor de JVM con DB2, debe tener la versión más reciente de IBM Data Server Driver para JDBC y SQLJ. Para obtener más información acerca de los APAR necesarios, consulte los requisitos del sistema en http://www-01.ibm.com/support/docview.wss?uid=swg27020857.

Acerca de esta tarea

CICS proporciona un paquete middleware OSGi llamado `com.ibm.cics.db2.jcc.jar` para trabajar con los controladores de DB2 . Debe instalar el paquete middleware en la infraestructura OSGi de manera que las aplicaciones puedan acceder a DB2.

Procedimiento

1. Abra el perfil del servidor de JVM correspondiente al servidor de JVM adecuado. Puede utilizar la perspectiva z/OS de CICS Explorer para copiar, editar y guardar los perfiles de JVM.
2. Añada la ubicación del directorio `lib` para el controlador de DB2 apropiado para la opción`LIBPATH_SUFFIX` .
3. Añada la propiedad del sistema de JVM **`-Dcom.ibm.cics.db2.jcc.jdbc.home`** al perfil y especifique la ubicación del controlador deDB2 .

4. Añada el paquete middleware com.ibm.cics.db2.jcc.jar a la opción OSGI_BUNDLES .
5. Guarde los cambios.
6. Si está actualizando un servidor de JVM existente, inhabilite y habilite el recurso JVMSERVER. En caso contrario, cree un recurso JVMSERVER . CICS inicia la infraestructura OSGi e instala el paquete middleware.

Ejemplo

El siguiente fragmento muestra un perfil de JVM de ejemplo con las opciones necesarias:

```
#*********************************************************************
#
#                          Parámetros necesarios
#                          ---------------------
#
# Al utilizar un servidor de JVM, el conjunto de opciones de CICS que se soportan
JAVA_HOME=/usr/lpp/java/J6.0.1_64
WORK_DIR=.
LIBPATH_SUFFIX=/usr/lpp/db2910/lib
...
#*********************************************************************
#
#                     Parámetros específicos de servidor de JVM
#                     -----------------------------
#
OSGI_BUNDLES=/usr/lpp/cicsts42/lib/com.ibm.cics.db2.jcc.jar
#
OSGI_FRAMEWORK_TIMEOUT=60
#
#*********************************************************************
#
#                               Opciones de JVM
#                               -----------
# La siguiente opción define la política de recogida de basura.
#
-Xgcpolicy:gencon#
#*********************************************************************
#
#                 Definición de propiedades del sistema de JVM del usuario
#                 --------------------------------
#
# -Dcom.ibm.cics.some.property=algún_valor
#Djdbc.drivers=com.ibm.db2.jcc.DB2Driver
-Dcom.ibm.cics.db2.jcc.jdbc.home=/usr/lpp/db2910
#
#*********************************************************************
#
#              Variables de entorno de UNIX System Services
#              ----------------------------------------
#
JAVA_DUMP_OPTS="ONANYSIGNAL(JAVADUMP,SYSDUMP),ONINTERRUPT(NONE)"
#
#
```

Instalación de paquetes OSGi en un servidor de JVM

Para desplegar una aplicación Java en un servidor de JVM es necesario instalar los paquetes OSGi correspondientes a dicha aplicación en la infraestructura OSGi del servidor de JVM de destino.

Antes de empezar

Un paquete de CICS que contenga los paquetes OSGi correspondientes a la aplicación se debe desplegar en un zFS. El servidor de JVM de destino debe estar ejecutándose en la región CICS.

Acerca de esta tarea

Un paquete CICS puede contener uno o más paquetes y servicios OSGi. Como el paquete CICS es la unidad de despliegue, todos los paquetes y servicios OSGi se gestionan juntos como parte del recurso BUNDLE. La infraestructura OSGi también gestiona el ciclo de vida de los paquetes y servicios OSGi, lo que incluye la gestión de dependencias y el mantenimiento de versiones.

Como práctica recomendada, asegúrese de que todos los paquetes OSGi que componen una aplicación Java se despliegan en el mismo paquete CICS. Si se utiliza este método, se puede gestionar la aplicación como una única entidad mediante el recurso BUNDLE. Si hay dependencias entre paquetes OSGi, despliéguelos en el mismo paquete CICS. Cuando se instala el recurso BUNDLE de CICS, CICS se asegura de que todas las dependencias entre los paquetes OSGi se resuelven.

Si tiene dependencias en un paquete OSGi que contenga una biblioteca de código común, la práctica recomendada es crear un paquete CICS independiente para la biblioteca. En este caso, es importante instalar primero el recurso BUNDLE de CICS que contenga la biblioteca. Si se instala la aplicación Java antes que el paquete CICS del que depende, la infraestructura OSGi no es capaz de resolver las dependencias de la aplicación Java.

Procedimiento

1. Compruebe el servidor de JVM de destino en el paquete de CICS para asegurarse de que existe un servidor de JVM con dicho nombre en la región CICS.
2. Cree un recurso BUNDLE que especifique el directorio del paquete en el zFS:
   a. Pulse en **Definiciones** > **Definiciones de paquete** en la barra de menús de CICS Explorer para abrir la vista de definiciones de paquetes.
   b. Pulse con el botón derecho del ratón en cualquier lugar de la vista y pulse en **New** (Nuevo) para abrir el asistente para nueva definición de paquete. Escriba los detalles del recurso BUNDLE en los campos del asistente.
   c. Instale el recurso BUNDLE. Puede instalar el recurso en estado habilitado o inhabilitado:
      - Si instala el recurso en un estado DISABLED (inhabilitado), CICS instala los paquetes OSGi en la infraestructura y resuelve las dependencias, pero no intenta lanzar los paquetes.
      - Si instala el recurso en un estado ENABLED (habilitado), CICS instala los paquetes OSGi, resuelve las dependencias y lanza los paquetes OSGi. Si el paquete OSGi contiene un activador de paquete diferido, la infraestructura OSGi no intenta lanzar el paquete hasta que antes lo llame otro paquete OSGi.
3. Opcional: Habilite el recurso BUNDLE para lanzar los paquetes OSGi en la infraestructura si el recurso no está ya en un estado ENABLED.
4. Pulse en **Operaciones** > **Paquetes** en la barra de menús de CICS Explorer para abrir la vista Paquetes. Compruebe el estado del recurso BUNDLE.

- Si el recurso BUNDLE está en estado ENABLED (habilitado), CICS pudo instalar todos los recursos en el paquete de forma correcta.
- Si el recurso BUNDLE está en estado DISABLED (inhabilitado), CICS no pudo instalar uno o más recursos en el paquete de forma correcta.

Si el recurso BUNDLE no se pudo instalar en el estado habilitado, compruebe las partes del paquete del recurso BUNDLE. Si cualquiera de las partes del paquete está en estado UNUSABLE (inutilizable), CICS no pudo crear los paquetes OSGi. Normalmente, este estado indica que hay un problema con el paquete CICS en zFS. Debe descartar el recurso BUNDLE, arreglar el problema y luego volver a instalar el recurso BUNDLE.

5. Pulse en **Operaciones** > **Java** > **Paquetes OSGi** en la barra de menús deCICS Explorer para abrir la vista Paquetes OSGI. Compruebe el estado de los paquetes y servicios OSGi instalados en la infraestructura OSGi. La siguiente tabla resume los estados:

BUNDLE	BUNDLEPART	OSGIBUNDLE	OSGISERVICE
ENABLED	ENABLING	INSTALLED	N/A
	ENABLED	STARTING	N/A
		ACTIVE	ACTIVE
			INACTIVE
DISABLED	DISABLING	STOPPING	N/A
	DISABLED	RESOLVED	N/A
	UNUSABLE	N/A	N/A

- Si el paquete OSGi está en estado STARTING (lanzándose), se ha llamado al activador de paquete pero este aún no se ha devuelto. Si el paquete OSGi tiene una política de activación diferida, el paquete permanece en este estado hasta que es llamado en la infraestructura OSGi.
- Si los paquetes OSGi y los servicios OSGi están activos, la aplicación Java está preparada.
- Si el servicio OSGi está inactivo, CICS detectó que ya existe un servicio OSGi con ese nombre en la infraestructura OSGi.

Resultados

El recurso BUNDLE está habilitado, los paquetes OSGi se han instalado correctamente en la infraestructura OSGi y todos los servicios OSGi están activos. Los paquetes y los servicios OSGi están disponibles para otros paquetes en la infraestructura.

Qué hacer a continuación

Puede hacer que la aplicación Java esté disponible para otras aplicaciones CICS fuera de la infraestructura OSGi.

Llamada a una aplicación Java en un servidor de JVM

Para llamar a una aplicación Java que se ejecute en un servidor de JVM desde otra aplicación de CICS, debe utilizar un servicio OSGi que esté activo en la infraestructura OSGi.

Acerca de esta tarea

Un servicio OSGi es una interfaz bien definida que está registrada en la la infraestructura OSGi para un paquete OSGi. Otros paquetes OSGi y aplicaciones remotas utilizan el servicio OSGi para llamar al código de aplicación que está empaquetado en el paquete OSGi. Un paquete OSGi puede tener más de un servicio OSGi.

La infraestructura OSGi gestiona las invocaciones de servicios para paquetes OSGi que están instalados en la misma infraestructura. Para llamar a una aplicación Java desde una aplicación de CICS que esté fuera la infraestructura OSGi, utilice el servicio OSGi adecuado para el paquete OSGi.

Procedimiento

1. Determine el nombre simbólico del servicio OSGi activo que quiere utilizar en la infraestructura OSGi. Haga clic en **Operations** > **Java** > **OSGi Services** (Operaciones > Java > Servicios OSGi) en CICS Explorer para indicar los servicios OSGi que están activos.
2. Cree un recurso PROGRAM para representar el servicio OSGi para otras aplicaciones CICS:
   a. En el atributo JVM, especifique `YES` para indicar que el programa es un programa Java.
   b. En el atributo JVMCLASS, especifique el nombre simbólico del servicio OSGi. Este valor distingue entre mayúsculas y minúsculas.
   c. En el atributo JVMSERVER, especifique el nombre del atributo JVMSERVER en el que se ejecuta el servicio OSGi.
3. Puede llamar a la aplicación Java de distintas formas:
   - Utilice una solicitud 3270 o **`EXEC CICS START`** que especifique un identificador de transacción. Cree un recurso TRANSACTION que defina el recurso PROGRAM para el servicio OSGi.
   - Utilice una solicitud de **`EXEC CICS LINK`**, una llamada a ECI o una llamada a EXCI. Indique el recurso PROGRAM para el servicio OSGi al codificar la solicitud.
   - Utilice una entrada en una tabla de lista de programas (PLT). Indique el recurso PROGRAM para el servicio OSGi.

Resultados

Ha creado un recurso PROGRAM para hacer que un paquete OSGi esté disponible para otras aplicaciones de CICS. Cuando se llama al servicio OSGi, CICS ejecuta la solicitud en el servidor de JVM de destino. Si el servicio OSGi está registrado como activo, el programa Java se ejecuta correctamente. Si el servicio OSGi no está registrado o está inactivo, se devuelve un error al programa de llamada.

Habilitación del gestor de seguridad Java

De forma predeterminada, las aplicaciones Java no tienen restricciones de seguridad sobre las actividades solicitadas de la API Java. Para utilizar la seguridad Java con el fin de proteger una aplicación Java y que esta no realice acciones potencialmente inseguras, puede habilitar un gestor de seguridad para la JVM en la que se ejecuta dicha aplicación.

Acerca de esta tarea

El gestor de seguridad impone una política de seguridad, que es un conjunto de permisos (privilegios de acceso del sistema) que se asigna a orígenes de código. Con la plataforma Java se proporciona un archivo de política predeterminado. Sin embargo, para permitir que las aplicaciones Java se ejecuten correctamente en CICS cuando la seguridad Java está activa, debe especificar un archivo de política adicional que proporcione a CICS los permisos necesarios para ejecutar la aplicación.

Debe especificar este archivo de política adicional para cada tipo de JVM que tenga un gestor de seguridad habilitado. CICS proporciona algunos ejemplos que puede utilizar para crear sus propias políticas.

Procedimiento

- Para aplicaciones que se ejecuten en la infraestructura OSGi de un servidor de JVM:
  1. Cree un proyecto de plug-in en CICS Explorer SDK y seleccione el programa de ejemplo del agente de seguridad de OSGi proporcionado. Este ejemplo crea un paquete middleware de OSGi llamado `com.ibm.cics.server.examples.security` en el proyecto que contiene un perfil de seguridad. Este perfil se aplica a todos los paquete OSGi de la infraestructura en la que está instalado.
  2. En el proyecto, seleccione el archivo `example.permissions` para editar los permisos para la política de seguridad. Este archivo contiene permisos específicos para ejecutar aplicaciones en un servidor de JVM, incluyendo una comprobación para garantizar que las aplicaciones no utilizan en método System.exit() .
  3. Despliegue el paquete OSGi a un directorio adecuado en zFS. CICS debe poseer acceso de lectura y ejecución a este directorio.
  4. Cree un archivo de política para proporcionar todos los permisos a Java Launcher. Se proporciona una política de ejemplo llamada `all.policy` en el proyecto de plug-in. No se incluye en el paquete middleware, pero puede copiarlo a un directorio adecuado en zFS. El archivo de política contiene los siguientes permisos:
     ```
     grant {
     permission java.security.AllPermission;
     };
     ```
  5. Edite el perfil de JVM correspondiente al servidor de JVM para añadir el paquete OSGi a la opción `OSGI_BUNDLES` antes de cualquier otro paquete:
     ```
     OSGI_BUNDLES=/u/bundles/com.ibm.cics.server.examples.security_1.0.0.jar,
     /usr/lpp/cicsts42/lib/com.ibm.cics.db2.jcc.jar
     ```
  6. Añada la siguiente variable de entorno de Java al perfil de JVM para habilitar la seguridad en la infraestructura OSGi:
     ```
     org.osgi.framework.security=osgi
     ```
  7. Añada la siguiente propiedad del sistema de seguridad de Java al perfil de JVM para especificar la política de seguridad:
     ```
     -Djava.security.policy=/u/policies/all.policy
     ```
  8. Guarde los cambios y habilite el recurso JVMSERVER para instalar el paquete middleware en el servidor de JVM.
- Para aplicaciones que se ejecutan en una JVM en agrupación, utilice el archivo `dfhjejbpl.policy` para implementar la política de seguridad.

1. Cree un archivo de política para la aplicación en `/usr/lpp/java/J6.0.1_64/lib/security/`, donde `java/J6.0.1_64` es la ubicación para el IBM 64 bits SDK para z/OS, Java Technology Edition.

   El gestor de seguridad utiliza siempre el archivo de política predeterminado `java.policy` que se incluye en este directorio. Si desea que una aplicación utilice JDBC o SQLJ, cree un archivo de política para otorgar permisos al controlador JDBC. Debe utilizar el controlador JDBC 2.0 con seguridad Java.
2. Habilite el gestor de seguridad añadiendo la propiedad del sistema **`-Djava.security.manager`** al perfil de JVM. Utilice uno de los siguientes formatos:
   - `-Djava.security.manager=default`
   - `-Djava.security.manager=""`
   - `-Djava.security.manager=`
3. Especifique sus archivos de política añadiendo la propiedad del sistema **`-Djava.security.policy`** al perfil de JVM. El gestor de seguridad utiliza cualquier política definida en esta propiedad además de la política de seguridad predeterminado.

Resultados

Cuando se llama a la aplicación Java, la JVM determina el origen del código para la clase y consulta la política de seguridad antes de otorgar a dicha clase los permisos adecuados.

Configuración de JVM en agrupación

El entorno de JVM en agrupación es necesario para ejecutar Enterprise Java Beans, CORBA y aplicaciones Java que no sean de enhebramiento seguro. Debe configurar los perfiles de JVM y los recursos de CICS que se proporcionan. Como opción, puede ejecutar la aplicación de ejemplo Hello World para comprobar si el entorno está configurado correctamente.

Acerca de esta tarea

Procedimiento

1. Copie los ejemplos provistos DFHJVMPR y DFHJVMCD desde su ubicación de instalación al directorio de z/OS UNIX que se especifica en el parámetro de inicialización del sistema **`JVMPROFILEDIR`**. Trabajar con copias de los perfiles provistos garantiza que no se pierdan los cambios si los perfiles se actualizan al aplicar mantenimiento.
   - DFHJVMPR es el perfil provisto para la JVM en agrupación.
   - DFHJVMCD es el perfil provisto para para programas de sistema y la memoria caché de clase compartida.
2. Personalice los perfiles de JVM para editar las opciones que configuran la JVM cuando esta se lanza. Por ejemplo, cambie la cantidad de almacenamiento que hay disponible y aplique valores de seguridad. Las opciones se explican en “Perfiles de JVM: opciones y ejemplos” en la página 104.
3. Opcional: Compruebe la configuración de su entorno de JVM en agrupación mediante la aplicación de ejemplo HelloWorld de JCICS.
4. Cree los recursos de CICS y el perfil de JVM para permitir que una aplicación, Enterprise Java Bean o CORBA, utilice una JVM en agrupación. Puede utilizar el perfil predeterminado personalizado, DFHJVMPR, o puede crear su propio perfil.

Resultados

El entorno está configurado y ha creado los recursos de CICS para ejecutar una aplicación Java en una JVM en agrupación.

Personalización de DFHJVMCD

El perfil de JVM DFHJVMCD se reserva para que lo utilicen programas de sistema proporcionados por CICS, en concreto el programa procesador de solicitudes predeterminado, DFJIIRP, que es empleado por la transacción del procesador de solicitudes CIRP proporcionado por CICS. CICS también utiliza DFHJVMCD para inicializar y terminar la memoria caché de clase compartida para las JVM en agrupación.

Antes de empezar

DFHJVMCD se debe configurar correctamente para la región CICS, pero solo debe personalizarse cuando sea necesario. DFHJVMCD puede tener un archivo de propiedades de JVM asociado, pero es opcional.

Asegúrese de que trabaja con una copia de DFHJVMCD en el directorio de z/OS UNIX que especificara en el parámetro de inicialización del sistema **JVMPROFILEDIR** y no con el archivo original en su ubicación de instalación.

Acerca de esta tarea

Las opciones que se pueden modificar se indican en el texto de DFHJVMCD. No realice ningún otro cambio en los archivos.

Para obtener información detallada sobre las opciones de DFHJVMCD que se pueden cambiar y la finalidad de cambiarlas, consulte “Opciones para JVM en un entorno CICS” en la página 109 y “Propiedades del sistema de la JVM” en la página 118.

Procedimiento

1. Abra DFHJVMCD en un editor de texto estándar.
2. Si cuenta con una memoria caché de clase compartida en la región CICS, puede especificar que las JVM en agrupación que se crearan con el perfil DFHJVMCD utilicen dicha memoria caché de clase compartida. Cambie la opción CLASSCACHE a CLASSCACHE=YES. El valor predeterminado, CLASSCACHE=NO, significa que son JVM autónomas.
3. Cambie el valor de la opción JAVA_HOME si este no coincide con el directorio de instalación de IBM 64 bits SDK para z/OS, Java Technology Edition en z/OS UNIX.
4. Para cambiar el directorio de trabajo en z/OS UNIX que utilizan las JVM con el perfil DFHJVMCD, cambie la opción WORK_DIR para que especifique el directorio preferido.
5. Para cambiar los nombres de los archivos de z/OS UNIX que utilizar para `stderr`, `stdin` y `stdout`, cambie las opciones STDERR, STDIN y STDOUT.
6. Para utilizar una clase de redirección de salida que intercepte y redirija la salida y los mensajes desde la JVM, utilice la opción USEROUTPUTCLASS para especificar el nombre de dicha clase. No utilice esta opción en un entorno de producción.

7. Para ajustar el tamaño de almacenamiento dinámico para las JVM con el perfil DFHJVMCD y que se adapte mejor a las necesidades de sus aplicaciones, cambie las opciones **-Xms** o **-Xmx**.
8. Si dispone de aplicaciones que utilicen JDBC, añada las bibliotecas y los archivos DB2 necesarios, según se especifica en el perfil de ejemplo DFHJVMPR, a las opciones LIBPATH_SUFFIX y CLASSPATH_SUFFIX de DFHJVMCD.
9. Habilite el mecanismo de política de seguridad de Java (la propiedad del sistema **-Djava.security.policy**) si es necesario para la instalación.
10. Guarde los cambios en el perfil. Confirme que la versión personalizada de DFHJVMCD está en el directorio de z/OS UNIX que especificara en el parámetro de inicialización del sistema **JVMPROFILEDIR**.

Qué hacer a continuación

No especifique DFHJVMCD en los recursos PROGRAM que configure para sus propias aplicaciones Java. Tal vez desee realizar cambios de personalización similares en una copia del otro perfil de JVM de ejemplo suministrado por CICS, DFHJVMPR, para que lo utilicen sus aplicaciones.

Personalización de DFHJVMPR

DFHJVMPR es el perfil de JVM predeterminado para JVM en agrupación. Cualquier cambio que se realice a este perfil se aplica a todas las JVM en agrupación en las que el recurso PROGRAM no especifique otro perfil de JVM.

Antes de empezar

Antes de empezar, asegúrese de que trabaja con la copia del perfil de JVM incluida y no con el archivo original de su ubicación de instalación.

Acerca de esta tarea

Cuando personalice el perfil DFHJVMPR, asegúrese de que los cambios son adecuados para todas las aplicaciones Java que utilicen el perfil. Si desea añadir opciones para una aplicación Java que no se apliquen a las demás aplicaciones, cree un perfil de JVM basado en DFHJVMPR con un nombre distinto.

Procedimiento

1. Abra el perfil de JVM en un editor de texto estándar y edite las opciones. Utilice las listas de opciones que se muestran en "Perfiles de JVM: opciones y ejemplos" en la página 104. Cada parámetro o propiedad se especifica en una línea distinta, y el valor del parámetro o la propiedad se delimita mediante el final de la línea. Siga las reglas de codificación que aparecen en "Reglas para codificación de perfiles de JVM" en la página 106.

   Tal vez desee cambiar estas opciones clave para las JVM en agrupación:

   - Habilite la seguridad Java añadiendo un gestor de seguridad y una política al perfil. El mecanismo de política de seguridad de Java protege las aplicaciones Java para que no realicen acciones poco seguras. Puede añadir seguridad al perfil utilizando las propiedades del sistema **-Djava.security.manager** y **-Djava.security.policy**. Para obtener más información, consulte "Habilitación del gestor de seguridad Java" en la página 93.
   - Cambie la opción **-Xmx** en el perfil de JVM para ajustar la cantidad de almacenamiento disponible para la aplicación. Esta opción cambia el tamaño

del almacenamiento dinámico en la JVM. El valor predeterminado es de 16 MB, pero si cuenta con aplicaciones Java de gran tamaño, tal vez desee aumentar dicho valor.

- Cambie el umbral de tiempo de espera para la JVM, utilizando la opción `IDLE_TIMEOUT` en el perfil de JVM. El valor predeterminado es que una JVM inactiva pasa a ser elegible para terminación automática por CICS tras 30 minutos. Si prefiere mantener JVM sin usar disponibles durante un periodo más prolongado, puede especificar un umbral de tiempo de espera de hasta 7 días o definir que la JVM nunca exceda el tiempo de espera.
- Cambie el destino para los mensajes y la salida procedentes de la JVM. Puede modificar el nombre y la ubicación de los archivos `stdin`, `stdout` y `stderr` de los volcados de Java, y utilice símbolos para que estos archivos sean exclusivos para cada JVM. Durante el desarrollo de aplicaciones, puede redirigir los mensajes internos y la salida de la JVM desde aplicaciones Java utilizando la opción `USEROUTPUTCLASS` una clase en el perfil de JVM. "Control de la ubicación para las salidas de JVM stdout, stderr y de volcado" en la página 197 ofrece más información sobre los cambios que puede realizar.

2. Opcional: Si sus aplicaciones Java necesitan acceso a datos de DB2 utilizando JDBC, emplee la propiedad del sistema **`-Djdbc.drivers`**. Para obtener más información, consulte Uso de JDBC y SQLJ para acceder a datos de DB2 desde programas Java en la Guía DB2.
3. Guarde el perfil de JVM personalizado en el directorio de z/OS UNIX que se especifica mediante el parámetro de inicialización del sistema **`JVMPROFILEDIR`** para su región CICS. CICS carga los perfiles de JVM desde este directorio.
4. Confirme que CICS tiene acceso de lectura y grabación a z/OS UNIX para el perfil de JVM y para el directorio que lo contiene.
5. Si cuenta con JVM en ejecución que utilicen el perfil DFHJVMPR, saque de la fase la agrupación de JVM para el perfil. Todas las JVM existentes que utilizan DFHJVMPR se detienen y se inician. Las nuevas JVM utilizan la última versión del perfil de JVM.

Resultados

Ha personalizado el perfil de JVM de ejemplo para JVM en agrupación y se ha asegurado de que CICS cuenta con el acceso correcto al perfil en el directorio de z/OS UNIX.

Qué hacer a continuación

Puede configurar aplicaciones para utilizar el perfil de JVM personalizado y añadir las clases para la aplicación a la vía de acceso de clases. Consulte "Habilitación de una aplicación para que utilice una JVM en agrupación" en la página 101.

Creación de sus propios perfiles de JVM

Puede crear un perfil de JVM con un nombre de archivo distinto para una aplicación específica, o si no quiere tener que realizar actualizaciones en los perfiles de ejemplo personalizados.

Antes de empezar

Debe tener copias de los perfiles de ejemplo proporcionados en el directorio de z/OS UNIX que se especifica mediante el parámetro de inicialización del sistema **`JVMPROFILEDIR`** para su región CICS. CICS debe poseer acceso de lectura y ejecución a este directorio.

Acerca de esta tarea

Para minimizar la administración, copie y renombre siempre el perfil de ejemplo apropiado. Si la aplicación Java va a ejecutarse en un servidor de JVM, utilice el perfil DFHOSGI como base para las modificaciones. Si la aplicación Java va a ejecutarse en una JVM en agrupación, utilice el perfil DFHJVMPR como base para las modificaciones.

Procedimiento

1. Copie y renombre el perfil de ejemplo de JVM apropiado. No proporcione al perfil de JVM un nombre que empiece por DFH, porque estos caracteres están reservados para el uso de CICS. Los nombres de los perfiles de JVM distinguen entre mayúsculas y minúsculas. Para obtener más información sobre la denominación de perfiles, consulte "Perfiles de JVM" en la página 7.
2. Edite el perfil de JVM en un editor de texto estándar utilizando las listas de opciones de "Perfiles de JVM: opciones y ejemplos" en la página 104. Algunas opciones son específicas para servidores de JVM y otras son específicas para JVM en agrupación.

   Cada parámetro o propiedad se especifica en una línea distinta, y el valor del parámetro o la propiedad se delimita mediante el final de la línea. Siga las reglas de codificación que aparecen en "Reglas para codificación de perfiles de JVM" en la página 106.
3. Si desea habilitar la seguridad Java, especifique las opciones de seguridad y configure uno o más archivos de política de seguridad para definir propiedades para la JVM. Para obtener más detalles, consulte "Habilitación del gestor de seguridad Java" en la página 93.
4. Guarde el perfil de JVM en el directorio de z/OS UNIX que se especifica mediante el parámetro de inicialización del sistema **JVMPROFILEDIR** para su región CICS. CICS carga los perfiles de JVM desde este directorio.
5. Asegúrese de que CICS tiene acceso de lectura a z/OS UNIX para el perfil de JVM. Consulte "Concesión a las regiones CICS de acceso a directorios y archivos z/OS UNIX" en la página 83.

Resultados

Ha creado un perfil de JVM que cumple con los convenios de denominación y contiene las opciones correctas para la aplicación Java y el entorno de ejecución pretendido.

Qué hacer a continuación

Configure la aplicación para que utilice el perfil de JVM, lo que incluye la creación de recursos de CICS. Puede configurar un servidor de JVM, como se describe en "Configuración de un servidor de JVM" en la página 87, o una JVM en agrupación, como se describe en "Habilitación de una aplicación para que utilice una JVM en agrupación" en la página 101.

Comprobación de la configuración de JVM en agrupación con los ejemplos

Configure y ejecute los programas de ejemplo "Hello World" y "Hello CICS World" para verificar que el soporte del entorno de JVM en agrupación está configurado correctamente en la región CICS.

Antes de empezar

Antes de ejecutar los programas de ejemplo, asegúrese de haber completado las demás tareas de configuración descritas en Capítulo 4, “Configuración del soporte de Java”, en la página 81.

Acerca de esta tarea

Los programas de ejemplo de Java se proporcionan en el CICS Explorer SDK para ayudar a los desarrolladores de aplicaciones a iniciarse en el desarrollo de aplicaciones Java en CICS. Los archivos de origen y compilación de Java también se incluyen en z/OS UNIX durante la instalación de CICS si desea ejecutar los programas de ejemplo para verificar que el entorno de JVM en agrupación esté configurado correctamente.

Para configurar y ejecutar los programas proporcionados, debe definir variables de entorno en z/OS UNIX. Puede definir las variables en un perfil para z/OS UNIX mediante el mandato **export**, o puede escribir el mandato **export** manualmente cuando inicie sesión en z/OS UNIX.

Procedimiento

1. PATH es la vía de acceso de búsqueda de z/OS UNIX System Services. Defina la variable de entorno PATH:

   ```
   /usr/lpp/java/J6.0.1_64/bin
   ```

   La vía de acceso indica donde se instala IBM 64 bits SDK para z/OS, Java Technology Edition en z/OS UNIX. Puede utilizar el mandato **export** para añadir la vía de acceso del siguiente modo:

   ```
   export PATH=/usr/lpp/java/J6.0.1_64/bin:$PATH
   ```

2. CICS_HOME es el directorio de instalación para archivos de CICS Transaction Server para z/OS en z/OS UNIX System Services. Defina la variable de entorno CICS_HOME del siguiente modo:

   ```
   /usr/lpp/cicsts/vía_acceso_cics
   ```

   El valor de *vía_acceso_cics* se define mediante el parámetro de instalación **USSDIR** cuando se instala CICS TS. `cicsts42` es el valor predeterminado. Puede utilizar el mandato export para definir el prefijo del directorio del siguiente modo:

   ```
   export CICS_HOME=/usr/lpp/cicsts/cicsts42
   ```

   El directorio `$CICS_HOME/samples/dfjcics` contiene los archivos make.

   El directorio `$CICS_HOME/samples/dfjcics/examples` contiene los archivos de origen Java.

3. JAVA_HOME especifica la vía de acceso a los subdirectorios de IBM 64 bits SDK para z/OS, Java Technology Edition. Defina la variable de entorno JAVA_HOME del siguiente modo:

   ```
   /usr/lpp/java/ubicación_Java/
   ```

   ubicación_Java es donde se instala IBM 64 bits SDK para z/OS, Java Technology Edition en z/OS UNIX. El valor predeterminado es `java/J6.0.1_64/`.

4. Compilar los programas de ejemplo de Java:
   a. En el directorio `samples/dfjcics` , escriba `make jvm` para compilar todos los programas de ejemplo. Los archivos make llaman a `javac` y almacenan los

archivos de salida en el directorio *$CICS_HOME*/samples/dfjcics/examples/nombre_ejemplo de z/OS UNIX, donde nombre_ejemplo es el nombre del programa de ejemplo.

b. Compile y convierta los programas C proporcionados que se encuentran en SDFHSAMP:
   - DFH$LCCA
   - DFH$JSAM
   - DFH$LCCC

   Estos programas se enlazan mediante Program Control y uno de los programas de ejemplo de Java "Hello World". DFH$LCCA y DFH$JSAM son programas de CICS estándar que podrían estar escritos en cualquiera de los lenguajes soportados por CICS. Si no dispone de un compilador de C, puede escribir versiones COBOL de los programas incluidos y utilizarlas en lugar de las versiones C provistas.

c. Enlace los programas en DFHRPL o en una concatenación de biblioteca dinámica.

5. Añada la serie /usr/lpp/cicsts/cicsts42/samples/dfjcics a la opciónCLASSPATH_SUFFIX en el perfil de JVM predeterminado DFHJVMPR. /usr/lpp/cicsts/cicsts42 es el valor del parámetro de inicialización del sistema **USSHOME**.
6. Ejecute los programas de ejemplo de Hello World mediante los pasos descritos en "Ejecución de los ejemplos de Hello World" en la página 40.

Resultados

Los programas de ejemplo de Hello World se han ejecutado correctamente.

Qué hacer a continuación

Puede habilitar aplicaciones Java para ejecutarlas utilizando JVM en agrupación.

Habilitación de una aplicación para que utilice una JVM en agrupación

Si desea habilitar una aplicación Java para que utilice una JVM en agrupación, debe configurar los recursos de CICS para que ejecuten un programa Java en una JVM. También se debe definir dónde encontrar las clases para la aplicación.

Acerca de esta tarea

Utilice una JVM en agrupación cuando desee ejecutar una aplicación que no sea de enhebramiento seguro en una única JVM.

Procedimiento

1. Seleccione o cree un perfil de JVM adecuado para la aplicación Java. Puede copiar la aplicación de ejemplo DFHJVMPR que se proporciona. Todos los perfiles de JVM están ubicados en el directorio de z/OS UNIX que se especifica mediante el parámetro de inicialización del sistema **JVMPROFILEDIR**.
2. Edite el perfil de JVM para añadir las clases y las bibliotecas que sean necesarias para la aplicación. Puede utilizar cualquier editor de texto estándar. Utilice los dos puntos como separador entre vías de acceso. Para incluir saltos de línea, utilice una barra inclinada invertida y un espacio en blanco (\ ).
   a. Coloque las clases de aplicación en la vía de acceso de clase estándar. La vía de acceso de clase estándar se define mediante la opción **CLASSPATH_SUFFIX**.

No especifique el nombre de la propia clase ni el nombre del paquete en el perfil de JVM. Las opciones del perfil de JVM especifican la vía de acceso a la clase.

b. Si las clases no están en un paquete, incluya todos los subdirectorios en la vía de acceso de clases.

c. Si las clases o los paquetes están en archivos JAR, incluya el nombre del archivo JAR en la vía de acceso de clases.

Para obtener detalles sobre reglas para codificación de vías de acceso de clases y otros elementos en un perfil de JVM, consulte "Reglas para codificación de perfiles de JVM" en la página 106.

3. Añada los archivos nativos de biblioteca de enlace dinámico (DLL) de C que sean necesarios para la aplicación en la opción **LIBPATH_SUFFIX** del perfil de JVM.

   El middleware y las herramientas que proporcionan IBM u otros proveedores pueden requerir que se añadan archivos DLL a la vía de acceso a biblioteca; por ejemplo, se necesitan archivos DLL para utilizar los controladores JDBC de DB2. Es posible que también tenga código nativo asociado con una clase que haya escrito.

4. Guarde los perfiles de JVM en el directorio especificado por el parámetro de inicialización del sistema **JVMPROFILEDIR**.

5. Cree un recurso PROGRAM y defina los atributos Java adecuados. Al especificar valores para los atributos, asegúrese de utilizar las mayúsculas y minúsculas correctas para la clase de JVM y el perfil de JVM.

   a. En el atributo EXECKEY, especifique la clave de ejecución para el programa Java. El valor predeterminado para este atributo es USER y es adecuado para la mayoría de programas, Java ya que mejora la protección de almacenamiento. Sin embargo, si el programa forma parte de una transacción que especifique TASKDATAKEY(CICS), el programa se debe ejecutar en clave CICS.

   b. En el atributo JVM, especifique YES para indicar que el programa es un programa Java.

   c. En el atributo JVMCLASS, especifique el nombre de la clase principal en el programa Java. Si el programa se ha compilado como un paquete, especifique el nombre completo, que es el nombre de clase Java cualificado por el nombre del paquete, con el punto (.) utilizado como separador.

      Por ejemplo, el paquete example.HelloWorld contiene la clase HelloCICSWorld; en este caso, el nombre de clase completo es example.HelloWorld.HelloCICSWorld. Si el programa no se ha compilado como un paquete, especifique el nombre de clase sin ningún calificador.

   d. En el atributo JVMPROFILE, especifique el nombre del perfil que editara para incluir las clases de aplicación.

Resultados

El perfil de JVM y el recurso PROGRAM están disponibles en la región CICS para soportar la aplicación Java en ejecución. Cuando una aplicación realiza una solicitud para ejecutar un programa Java, puede realizarla de varias formas:

- Una solicitud 3270 o **EXEC CICS START** que especifique un identificador de transacción.
- Una solicitud **EXEC CICS LINK** o una llamada ECI o EXCI que da nombre al programa Java directamente.
- Una entrada en una tabla de lista de programas (PLT).

En el caso de solicitudes de **EXEC CICS LINK** o de llamadas de ECI o EXCI, y en el de entradas en una tabla de lista de programas, CICS recibe el nombre del recurso PROGRAM directamente. Sin embargo, en el caso de solicitudes 3270 o START, CICS determina el recurso PROGRAM utilizando el identificador de transacción.

Qué hacer a continuación

Si el perfil de JVM especifica que la JVM utilice la memoria caché de clase compartida, debe asegurarse de que la memoria caché de clase se haya iniciado o tenga habilitado el inicio automático para que se ejecuten las aplicaciones Java. En "Inicio de la memoria caché de clase compartida" en la página 151 se indica cómo iniciar la memoria caché de clase compartida o habilitar el inicio automático.

Habilitar aplicaciones CORBA o enterprise bean para que utilicen una JVM

Para habilitar una aplicación CORBA o enterprise bean para que utilice una JVM en agrupación, debe actualizar el perfil de JVM y la vía de acceso de clases para la aplicación.

Acerca de esta tarea

Los objetos sin estado CORBA y los enterprise beans no cuentan con sus propios recursos PROGRAM. Una solicitud de método para un enterprise bean o un objeto sin estado CORBA implica una JVM, ya que el procesador de solicitudes que la gestiona se ejecuta en una JVM. Un procesador de solicitudes es un programa que gestiona la ejecución de una solicitud de IIOP, lo que incluye llamar al contenedor para que procese el método. Cuando CICS recibe la solicitud de método, esta se compara con un recurso REQUESTMODEL, encuentra el que mejor coincida con la solicitud y utiliza el identificador de transacción de dicho modelo de solicitud para determinar el recurso PROGRAM.

En ocasiones, las solicitudes de IIOP se procesan utilizando una transacción del procesador de solicitudes ya existente que ya tenga una JVM asignada. CICS solo mira el identificador de transacción en cualquier modelo de solicitud coincidente cuando es necesaria una nueva transacción del procesador de solicitudes.

Procedimiento

1. Identifique el perfil de JVM para el programa procesador de solicitudes que gestiona el objeto sin estado CORBA o el enterprise bean. El perfil de JVM se especifica en el recurso PROGRAM correspondiente al programa procesador de solicitudes. El programa procesador de solicitudes predeterminado es DFHJIIRP, y el perfil de JVM predeterminado para este programa es DFHJVMCD.
2. Solo en el caso de objetos sin estado CORBA, añada el archivo JAR para la aplicación a la opción CLASSPATH_SUFFIX del perfil de JVM para el programa procesador de solicitudes. Utilice un símbolo de dos puntos como separador entre las vías de acceso que especifique en una vía de acceso de clases. Para incluir saltos de línea, utilice una barra inclinada invertida y un espacio en blanco (\ ).

   No es necesario que añada los archivos JAR desplegados (DJAR) para los enterprise beans a la vía de acceso de clases.

3. Si los enterprise beans o la aplicación CORBA utilizan cualquier clase, como clases para programas de utilidad, que no esté incluida en el archivo JAR, incluya estas clases en la vía de acceso de clases que utiliza la JVM para el programa procesador de solicitudes.

Resultados

El perfil de JVM y las vías de acceso de clases están disponibles en la región CICS para soportar la aplicación Java en ejecución.

Perfiles de JVM: opciones y ejemplos

CICS proporciona perfiles de JVM de ejemplo que contienen una selección de opciones para las JVM de IBM que se utilizan en un entorno CICS. Algunas de estas opciones son específicas del entorno CICS y no se utilizan para las JVM en otros entornos. Otras opciones son opciones estándar o no estándar de Java que se pueden utilizar para los JVM de IBM en cualquier entorno.

Puede especificar cualquier opción de JVM o propiedad del sistema en un perfil de JVM y esta se pasa a la JVM. Los perfiles de JVM para servidores de JVM y agrupaciones de JVM son distintos, por lo que algunas opciones solo se pueden especificar para un tipo de entorno Java.

Puede definir las propiedades del sistema en un archivo de propiedades de JVM. Sin embargo, los archivos de propiedades de JVM solo se soportan para agrupaciones de JVM y con CICS no se proporcionan archivos de propiedades de ejemplo.

No existe ningún repositorio central de todas las opciones y propiedades del sistema para la JVM. Algunas fuentes de información que se pueden utilizar son las siguientes:

- La documentación sobre IBM 64 bits SDK para z/OS, Java Technology Edition, versión 6.
- El *IBM SDK Java Technology Edition Version 6 Supplement*, que está disponible en http://www-03.ibm.com/systems/z/os/zos/tools/java/products/sdk601_64.html. Este documento contiene información específica para IBM 64 bits SDK para z/OS, Java Technology Edition, versión 6.0.1.
- La Java Guía de diagnósticos. Esta guía documenta las propiedades del sistema utilizadas para el rastreo y la determinación de problemas de la JVM.

Las bibliotecas de clases Java incluyen otras propiedades del sistema, y las aplicaciones pueden tener sus propias propiedades del sistema. La documentación de IBM Java es la fuente primaria de información, mientras que la documentación de CICS es una fuente de información secundaria.

La tabla de resumen, Tabla 9 en la página 105, lista las opciones que se utilizan en los perfiles de JVM de ejemplo y qué opciones aplicar a los servidores de JVM y a las agrupaciones de JVM. La tabla también incluye algunas opciones adicionales que puede utilizar para completar las tareas descritas en la documentación de CICS. La tabla indica el valor predeterminado de cada opción si este no se especifica en los perfiles de JVM de ejemplo.

Tabla 9. Tabla de referencia de opciones JVM para los JVM en un entorno CICS

Opción	Valor predet.	Servidor de JVM	Agrupación de JVM	Comentarios
Tipo de JVM				
CLASSCACHE	NO	No está soportada.	Soportada.	YES hace que JVM utilice la memoria caché de clase compartida, NO hace que no la utilice.
REUSE	YES	No está soportada.	Soportada.	YES hace que sea una JVM continua, NO hace que sea una JVM de uso único
Directorios				
JAVA_HOME	Ninguno	Soportada.	Soportada.	Necesaria, los perfiles de ejemplo incluyen este directorio
WORK_DIR	/tmp	Soportada.	Soportada.	
Vías de acceso				
CLASSPATH_PREFIX	Ninguno	No está soportada.	Soportada.	
CLASSPATH_SUFFIX	Ninguno	No está soportada.	Soportada.	
LIBPATH_PREFIX	Ninguno	Soportada.	Soportada.	
LIBPATH_SUFFIX	Ninguno	Soportada.	Soportada.	
OSGI_BUNDLES	Ninguno	Soportada.	No está soportada.	Defina si desea utilizar paquetes de middleware en un servidor de JVM.
Umbral de tiempo de espera excedido				
IDLE_TIMEOUT	30 minutos	No está soportada.	Soportada.	Se aplica únicamente a JVM en agrupación continua.
OSGI_FRAMEWORK_TIMEOUT	60 segundos	Soportada.	No está soportada.	
Valores y recursos adicionales para la JVM				
JVMPROPS	Ninguno	No está soportada.	Soportada.	Defina únicamente si utiliza un archivo de propiedades de JVM.
INVOKE_DFHJVMAT	NO	No está soportada.	Soportada.	Se aplica únicamente a JVM en agrupación de un solo uso.
Tamaños de almacenamiento dinámico para almacenamiento				
-Xms		Soportada.	Soportada.	Para obtener información sobre el valor predeterminado de **-Xms**, consulte la información de referencia en Default settings for the JVM

Tabla 9. Tabla de referencia de opciones JVM para los JVM en un entorno CICS (continuación)

Opción	Valor predet.	Servidor de JVM	Agrupación de JVM	Comentarios
-Xmx		Soportada.	Soportada.	Para obtener información sobre el valor predeterminado de **-Xmx**, consulte la información de referencia en Default settings for the JVM
Umbral de recogida de basura				
GC_HEAP_THRESHOLD	85%	No está soportada.	Soportada.	Se aplica únicamente a JVM en agrupación continua.
Salida de la JVM				
JVMTRACE	`dfhjvmtrc`	Soportada.	No está soportada.	
LEHEAPSTATS	NO	No está soportada.	Soportada.	
STDERR	`dfhjvmerr`	Soportada.	Soportada.	
STDIN	`dfhjvmin`	Soportada.	Soportada.	
STDOUT	`dfhjvmout`	Soportada.	Soportada.	
USEROUTPUTCLASS	Ninguno	Soportada.	Soportada.	Defina únicamente en un entorno de desarrollo.
Determinación de problemas y depuración de aplicación				
JAVA_DUMP_OPTS	YES	Soportada.	Soportada.	
-Xdebug	NO	Soportada.	Soportada.	
PRINT_JVM_OPTIONS	NO	Soportada.	Soportada.	Definida en YES solo de forma temporal

Variables de entorno de z/OS UNIX System Services

Además de las opciones de JVM y de las propiedades del sistema que se utilizan para construir la JVM, puede especificar variables de entorno de z/OS UNIX System Services en un perfil de JVM. Cualquier par de nombre y valor en un perfil de JVM que no se reconozca como opción de JVM o como propiedad del sistema se trata como una variable de entorno de z/OS UNIX System Services y se exporta. Las variables de entorno de z/OS UNIX System Services especificadas en un perfil de JVM solo se aplican a JVM creadas con dicho perfil.

Las opciones `JAVA_DUMP_OPTS` y `JAVA_DUMP_TDUMP_PATTERN` de los perfiles de JVM de ejemplo son variables de entorno de z/OS UNIX System Services. Otro ejemplo es la variable de entorno `TZ`, que se puede especificar para cambiar el huso horario de la JVM.

Las variables de entorno de z/OS UNIX System Services únicamente se pueden especificar en un perfil de JVM.

Reglas para codificación de perfiles de JVM

Puede editar los perfiles de JVM con cualquier editor de texto estándar. Siga estas reglas para la codificación de sus perfiles de JVM.

- El nombre de un perfil de JVM puede tener hasta 8 caracteres de longitud. El nombre de un archivo de propiedades puede tener cualquier longitud, pero, para facilitar su uso, generalmente es un nombre abreviado con alguna similitud con el nombre del perfil de JVM al que hace referencia.
- El nombre de un perfil de JVM o de un archivo de propiedades de JVM puede ser cualquier nombre que sea válido para un archivo z/OS UNIX System Services. No utilice un nombre que empiece por DFH, porque estos caracteres están reservados para que los utilice CICS.
- Como los perfiles de JVM y los archivo de propiedades de JVM son archivos UNIX, las mayúsculas y minúsculas son importantes. Al especificar el nombre en CICS, debe escribirlo utilizando la misma combinación de mayúsculas y minúsculas presente en el nombre de archivo z/OS UNIX.
- No utilice comillas al especificar valores para directorios en un perfil de JVM.
- Los paneles de CEDA aceptan entrada en mayúsculas y minúsculas para el campo JVMPROFILE, independientemente de los valores UCTRAN de su terminal. Sin embargo, debe escribir el nombre de un perfil de JVM en mayúsculas y minúsculas cuando utilice CEDA desde la línea de mandatos o cuando utilice otra transacción de CICS. Asegúrese de que el terminal está configurado correctamente, con la conversión de mayúsculas suprimida. Puede utilizar la transacción CEOT proporcionada para modificar el estado de conversión de mayúsculas (UCTRAN) para su propio terminal, únicamente para la sesión actual.

Siga estas reglas para la codificación de opciones de JVM o propiedad del sistema.

Diferenciación entre mayúsculas y minúsculas
Todas las palabras clave y los operandos de parámetros distinguen entre mayúsculas y minúsculas y deben especificarse exactamente como se muestra en “Opciones para JVM en un entorno CICS” en la página 109 y en “Propiedades del sistema de la JVM” en la página 118.

Carácter de separador de vía de acceso de clases
Utilice el carácter **:** (dos puntos) para separar las vías de acceso de directorio que se especifiquen en una opción de vía de acceso de clases, como CLASSPATH_SUFFIX.

Continuación
Para las opciones de JVM o propiedades del sistema, el valor se delimita mediante el final de la línea en el archivo de texto. Si un valor que está escribiendo o editando es demasiado largo para una ventana del editor, puede interrumpir la línea para evitar el desplazamiento. Para continuar en la siguiente línea, termine la línea actual con el carácter de barra inclinada invertida y un carácter de continuación en blanco, como en este ejemplo:

```
CLASSPATH_SUFFIX=/u/example/pathToJarOrZipFile/jarfile.jar:\
/u/example/pathToRootDirectoryForClasses
```

Comentarios
Para añadir comentarios o desactivar una opción mediante comentarios en lugar de suprimirla, comience cada línea del comentario con un símbolo #. Las línea de comentarios se ignoran cuando el lanzador de JVM lee el archivo.

Las líneas en blanco también se ignoran. Puede utilizar líneas en blanco como separador entre opciones o grupos de opciones.

Secuencias de escape de carácter
En una serie de elementos de propiedad, puede codificar las secuencias de escape que se muestran en la Tabla 10 en la página 108

Tabla 10. Secuencias de escape

Secuencia de escape	Valor de carácter
\b	Retroceso
\t	Tabulador horizontal
\n	Nueva línea
\r	Retorno de carro
\"	Comillas
\'	Comilla simple
\\	Barra inclinada invertida
xxx	El carácter que se corresponde con el valor octal *xxx*, donde *xxx* está entre los valores 000 y 377
\uxxxx	El carácter Unicode con la codificación *xxxx*, donde *xxxx* son entre 1 y 4 dígitos hexadecimales. (Consulte la nota para obtener más información.)

Nota: Los escapes \u Unicode son diferentes de los otros tipos de escape. Las secuencias de escape Unicode se procesan antes que las demás secuencias de escape descritas en Tabla 10. Un escape Unicode es una manera alternativa de representa un carácter que es posible que no se pueda visualizar en sistemas que no sean Unicode. Sin embargo, los escapes de carácter pueden representar caracteres especiales de un modo que impide la interpretación habitual de dichos caracteres.

Varias instancias de opciones
Puede utilizar cada opción solo una vez en un mismo perfil de JVM. Si se incluye más de una instancia de la misma opción en un perfil de JVM, se utiliza el valor para la última opción encontrada y se ignoran los valores anteriores.

Tamaños de almacenamiento
Al especificar opciones relacionadas con almacenamiento en un perfil de JVM, especifique tamaños de almacenamiento en múltiplos de 1024 bytes. Utilice la letra K para indicar KB, la letra M para indicar MB y la letra G para indicar GB. Por ejemplo, para especificar 6 291 456 bytes como el tamaño inicial del almacenamiento dinámico, codifique **-Xms** de uno de estos modos:

```
-Xms6144K
-Xms6M
```

Validación de las opciones de perfil de JVM

CICS lleva a cabo distintas comprobaciones de opciones clave especificadas en los perfiles de JVM siempre que se inician las JVM. Estas comprobaciones permiten una detección temprana de los problemas en la configuración de la JVM.

CICS lleva a cabo comprobaciones relacionadas con las siguientes opciones de perfil de JVM:

CLASSPATH_PREFIX, CLASSPATH_SUFFIX
En el caso de perfiles de servidor de JVM, CICS comprueba que estas opciones no se encuentren presentes en el perfil. Si se especifica cualquiera de estas opciones en el perfil, la infraestructura OSGi del servidor de JVM no se puede iniciar. El recurso JVMSERVER no se puede habilitar y CICS emite el mensaje de error DFHSJ0210.

JAVA_HOME
CICS comprueba los siguientes puntos para este directorio:

- El directorio existe en z/OS UNIX.
- CICS cuenta al menos con permiso de lectura para acceder al directorio.
- El archivo JDK_INSTALL_OK está presente en el directorio, lo que indica que la instalación de los archivos de IBM 64 bits SDK para z/OS, Java Technology Edition 6.0.1 en esta ubicación se ha completado.
- El número de release de Java en el archivo JDK_INSTALL_OK es una versión soportada por CICS.

Si se encuentra algún problema, CICS emite un mensaje de error y no lanza la JVM.

Opciones de vía de acceso de clases en desuso: LIBPATH, CICS_HOME, CLASSPATH, TMPREFIX y TMSUFFIX
Se emite un mensaje de aviso en el inicio de la JVM si cuenta con una o más de estas opciones en un perfil de JVM en agrupación. No utilice estas en perfiles de JVM. Estos mensajes recomiendan la opción correcta que utilizar en su lugar.

OSGI_BUNDLES
En el caso de perfiles de servidor de JVM, CICS comprueba que los archivos JAR especificados sean paquetes OSGi. CICS también comprueba que los paquetes de middleware estén delimitados correctamente y cuenten con los separadores correctos.

Opciones para JVM en un entorno CICS

Las opciones de un perfil de JVM las utilizan CICS, el IBM 64 bits SDK para z/OS, Java Technology Edition o z/OS UNIX System Services para lanzar JVM.

Cuando se especifiquen las opciones, asegúrese de seguir las reglas de codificación que se describen en “Reglas para codificación de perfiles de JVM” en la página 106. El formato de las opciones puede variar:

- Algunas opciones de una JVM toman la forma de una palabra clave y un valor separados por un signo =, como, por ejemplo, `LEHEAPSTATS=NO`.
- Algunas opciones se especifican con la opción seguida inmediatamente por el valor, sin ningún signo =, como, por ejemplo, `-Xms16M`.
- Cualquier opción que comience con un guión (-) es una opción estándar de Java o una opción no estándar de Java y se pasa a la JVM sin que CICS la analice.
- Puede especificar cualquier variable de entorno de z/OS UNIX System Services en un perfil de JVM. Cualquier par de nombre y valor en un perfil de JVM que no se reconozca como opción de JVM o como propiedad del sistema se trata como una variable de entorno de z/OS UNIX System Services y se exporta.
- Puede especificar propiedades del sistema de JVM, que comienzan por -D, en un perfil de JVM. Estas se indican por separado en “Propiedades del sistema de la JVM” en la página 118.

Para obtener información sobre las opciones de JVM **-Xmso**, **-Xiss** y **-Xss** y sobre todos los valores predeterminados, consulte la información de referencia en Default settings for the JVM.

Símbolos de perfil

Los siguientes símbolos se pueden utilizar en los valores de las opciones de un perfil de JVM, como se enseña en los perfiles de JVM de ejemplo.

&APPLID;
Si se utiliza este símbolo, el APPLID (identificador de aplicación) de la región CICS se sustituye en tiempo de ejecución. De este modo, se puede utilizar el mismo perfil o el mismo archivo de propiedades para todas las regiones y seguir teniendo directorios de trabajo o destinos de salida específicos de la región. El APPLID está siempre en mayúsculas. Puede utilizar el símbolo en las opciones `WORK_DIR`, `STDOUT`, `STDERR` y `JAVA_DUMP_TDUMP_PATTERN`.

&DATE;
Cuando se utiliza este símbolo, dicho símbolo se sustituye por la fecha actual con el formato *Daammdd* en tiempo de ejecución. Puede especificar el símbolo &DATE; para cualquiera tipo de salida de la JVM, incluidas las opciones `WORK_DIR`, `STDOUT`, `STDERR` y `JAVA_DUMP_TDUMP_PATTERN` .

&JVM_NUM;
Si se utiliza este símbolo, el número exclusivo de la JVM en agrupación se sustituye en tiempo de ejecución. Utilice este símbolo para crear archivos exclusivos de salida o de volcado para cada JVM. Puede utilizar el símbolo en las opciones `WORK_DIR`, STDOUT, `STDERR` y `JAVA_DUMP_TDUMP_PATTERN`. CICS puede modificar el número de la JVM para que sea conforme a los estándares de denominación de conjuntos de datos MVS para los TDUMP.

Este símbolo no se aplica a los servidores de JVM.

&JVMSERVER;
Si se utiliza este símbolo, el nombre del recurso JVMSERVER se sustituye en tiempo de ejecución. Utilice este símbolo para crear archivos exclusivos de salida o de volcado para cada servidor de JVM.

Este símbolo no se aplica a las JVM en agrupación.

&TIME;
Cuando se utiliza este símbolo, dicho símbolo se sustituye por la hora actual con el formato *Thhmmss* en tiempo de ejecución. Puede especificar el símbolo &TIME; para cualquiera tipo de salida de la JVM, incluidas las opciones `WORK_DIR`, `STDOUT`, `STDERR` y `JAVA_DUMP_TDUMP_PATTERN` .

Lista de opciones de JVM

En la lista de opciones siguiente todas las opciones se aplican a servidores de JVM y a JVM en agrupación, a no ser que se indique explícitamente. Cualquier valor predeterminado indicado para una opción es el valor que CICS utiliza cuando dicha opción no se especifica en un perfil de JVM. Algunos o todos los perfiles de JVM pueden especificar un valor que sea distinto al valor predeterminado.

CLASSCACHE={YES,NO}
Especifica si esta JVM va a utilizar la memoria caché de clase compartida. El valor predeterminado es NO.

Esta opción no se aplica a los servidores de JVM.

CLASSPATH_PREFIX, CLASSPATH_SUFFIX=*nombres_vía_acceso_clase*
La vía de acceso de clase estándar especifica vías de acceso del directorio, archivos de archivado Java y archivos comprimidos que debe buscar una JVM

en agrupación para clases y recursos de aplicación. Puede especificar las entradas en líneas separadas mediante una \ (barra inclinada invertida) al final de cada línea que se vaya a continuar.

CLASSPATH_PREFIX añade entradas de vía de acceso de clases al principio de la vía de acceso de clase estándar, y CLASSPATH_SUFFIX las añade al final de dicha vía de acceso de clase estándar.

Con la versión 6.0.1 del SDK, todas las clases de aplicación se colocan en la vía de acceso de clase estándar. En el caso de JVM en agrupación, todas las clases son elegibles para cargarlas en la memoria caché de clase compartida.

Utilice la opción `CLASSPATH_PREFIX` con cuidado. Las clases de `CLASSPATH_PREFIX` tienen preferencia sobre las clases del mismo nombre que proporcionan CICS y el tiempo de ejecución Java, lo que puede dar lugar a que se carguen las clases equivocadas.

CICS crea una vía de acceso de clases base para una JVM mediante los subdirectorios `/lib` de los directorios especificados por el parámetro de inicialización del sistema **USSHOME** y la opción `JAVA_HOME` del perfil de JVM. Esta vía de acceso de clases base contiene los archivos Java proporcionados por CICS y por la JVM. No es visible en el perfil de JVM. No vuelva a especificar estos archivos en las vías de acceso de clases del perfil de JVM.

Si define cualquier opción en el perfil de un servidor de JVM, la infraestructura OSGi no se lanza.

DISPLAY_JAVA_VERSION=
Si esta opción se define como YES, siempre que una aplicación lance una JVM, CICS graba el mensaje DFHSJ0901 en el registro MSGUSER, mostrando la versión y la compilación del IBM Software Developer Kit for z/OS, Java Technology Edition, que se utiliza.

GC_HEAP_THRESHOLD=
Especifica el límite de utilización del almacenamiento dinámico para el almacenamiento dinámico de la JVM. Cuando este porcentaje del almacenamiento en la parte activa del almacenamiento dinámico está en uso, CICS planifica una recogida de basura. CICS comprueba la utilización de almacenamiento dinámico tras la ejecución de cada programa Java. Si se alcanza el límite, la transacción de recogida de basura CJGC se planifica para ejecutarse en la JVM inmediatamente después de que finalice la utilización actual de la JVM.

El límite predeterminado de utilización del almacenamiento dinámico es 85 (85%). El mínimo es 50. El máximo si se desea que CICS planifique recogida de basura es 99. Si se especifica un límite de utilización del almacenamiento dinámico de 100, CICS nunca planifica recogidas de basura, y todas las recogidas de basura son resultado de anomalías de asignación mientras se ejecutan aplicaciones.

Esta opción no se aplica a los servidores de JVM ni a una JVM en agrupación de un solo uso.

IDLE_TIMEOUT={30|*número*}
Especifica el umbral de tiempo de espera, en minutos, para JVM en agrupación con este perfil de JVM. Si una JVM en agrupación está inactiva durante la cantidad de tiempo especificada, pasa a ser elegible para terminación automática. La siguiente vez que CICS comprueba las JVM desocupadas, si la JVM sigue inactiva, dicha JVM, así como su TCB J8 o J9, puede destruirse. CICS no detiene inmediatamente todas las JVM cuyo tiempo de espera ha finalizado; se detienen progresivamente con el tiempo.

El umbral de tiempo de espera predeterminado es 30 minutos y el máximo es 10.080 minutos (siete días). También puede especificar un umbral de tiempo de espera de cero, de forma que las JVM con ese perfil nunca se detengan automáticamente debido a inactividad. Las JVM con un umbral de tiempo de espera de cero se pueden detener si son seleccionadas para robo o no coincidencia, o si se restringe el almacenamiento MVS. Si se especifica un valor inaceptable, CICS utiliza en su lugar el valor predeterminado.

Esta opción no se aplica a servidores de JVM ni a una JVM en agrupación de un solo uso.

INVOKE_DFHJVMAT={NO|YES}
Especifica si se llama al módulo sustituible por el usuario, DFHJVMAT, antes de crear una JVM. DFHJVMAT puede utilizarse solo para JVM en agrupación de un solo uso; es decir, si la opción REUSE=NO se especifica en el perfil de JVM.

Esta opción no se aplica a servidores de JVM ni a una JVM en agrupación continua.

JAVA_DUMP_OPTS=
Una variable de entorno de z/OS UNIX System Services. Especifica un conjunto de opciones de volcado de Java que se utilizan para obtener información de diagnóstico en el caso de una terminación anómala en la JVM. Consulte la información sobre las opciones de volcado de Java en Dump agent environment variables.

JAVA_DUMP_TDUMP_PATTERN=
Una variable de entorno de z/OS UNIX System Services que especifica el nombre de archivo que utilizar para volcados de transacción (TDUMP) de la JVM. Los TDUMP de Java se graban en un destino de conjunto de datos en caso de una terminación anómala de JVM. Puede utilizar los símbolos &APPLID; (identificador de aplicación de región CICS) y &JVM_NUM; (número exclusivo de JVM) en este valor, como se muestra en los perfiles de JVM de ejemplo proporcionados, para crear nombres de archivo de volcado exclusivos para cada JVM.

Si se utiliza el símbolo &JVM_NUM; aquí, CICS puede modificar el número de JVM para que sea conforme a los estándares de denominación de conjuntos de datos de MVS. El número se formatea como un valor hexadecimal de 8 dígitos. Si el primer carácter es numérico, se debe modificar: 0 se cambia a G, 1 se cambia a H y así sucesivamente hasta el 9, que se cambia a P.

JAVA_HOME=/usr/lpp/java/J6.0.1_64/
Especifica la ubicación de instalación del IBM 64 bits SDK para z/OS, Java Technology Edition en z/OS UNIX. Esta ubicación contiene los subdirectorios y los archivos de archivado Java necesarios para el soporte de Java.

Los perfiles de JVM proporcionados contienen una vía de acceso que generó el parámetro **JAVADIR** en el trabajo de instalación DFHISTAR de CICS. El valor predeterminado para el parámetro **JAVADIR** es `java/J6.0.1_64/`, que es la ubicación de instalación predeterminada de IBM 64 bits SDK para z/OS, Java Technology Edition. Este valor produce un valor de `JAVA_HOME` en los perfiles de JVM de `/usr/lpp/java/J6.0.1_64/`.

JAVA_PIPELINE={YES,NO}
Añade los archivos de archivado Java necesarios a la vía de acceso de clases, de forma que un servidor de JVM pueda soportar el proceso de servicios web

en interconexiones SOAP basadas en Java. El valor predeterminado es NO. Si define este valor, el servidor de JVM se configura para soportar Axis2 en lugar de OSGi.

Esta opción no se aplica a las JVM en agrupación.

JVMPROPS=*vía_acceso/nombre_archivo*
Especifica la vía de acceso y el nombre de un archivo de propiedades de JVM opcional que es un archivo z/OS UNIX que se puede utilizar para contener propiedades del sistema para esta JVM. Para obtener más información sobre lo que se puede especificar en un archivo de propiedades de JVM, consulte "Propiedades del sistema de la JVM" en la página 118.

Esta opción no se aplica a los servidores de JVM.

JVMTRACE={*applid.jvmserver.***dfhjvmtrc|***nombre_archivo***}**
Especifica el nombre del archivo z/OS UNIX en el que se graba el rastreo de Java durante el inicio y la terminación de un servidor de JVM. Si no se especifica ningún valor, el rastreo se graba en el archivo *applid.jvmserver.*dfhjvmtrc. CICS crea automáticamente archivos de salida exclusivos para cada servidor de JVM mediante los símbolos &APPLID; y &JVMSERVER;. Este archivo se crea en el directorio especificado por la opción WORK_DIR.

Esta opción no se aplica a las JVM en agrupación.

LEHEAPSTATS={YES|NO}
Especifica si se van a recopilar estadísticas para la cantidad del almacenamiento dinámico de Language Environment que utiliza la JVM. El valor predeterminado es NO. Las estadísticas se notifican como el campo "Peak Language Environment heap storage used" (almacenamiento dinámico de Language Enviroment pico utilizado) en las estadísticas del perfil de JVM. La recopilación de estas estadísticas afecta al rendimiento de la JVM, de manera que debe especificar LEHEAPSTATS=YES solo mientras se ajusten los tamaños de almacenamiento dinámico para sus JVM. Para obtener más información, consulte . En un entorno de producción, especifique LEHEAPSTATS=NO.

Esta opción no se aplica a los servidores de JVM.

LIBPATH_PREFIX, LIBPATH_SUFFIX=*nombres_vía_de_acceso*
Especifica las vías de acceso de directorios en las que buscar archivos de biblioteca de enlace dinámico (DLL) nativas de C que utilice la JVM y que tengan la extensión .so en z/OS UNIX, incluidos los archivos necesarios para ejecutar la JVM y las bibliotecas nativas adicionales cargadas por el código de aplicación o por servicios.

La vía de acceso a biblioteca base para la JVM se crea automáticamente utilizando los directorios especificados por el parámetro de inicialización del sistema **USSHOME** y la opción JAVA_HOME en el perfil de JVM. La vía de acceso a biblioteca base no es visible en el perfil de JVM. Incluye todos los archivos DLL necesarios para ejecutar la JVM, así como las bibliotecas nativas que utiliza CICS.

Puede ampliar la vía de acceso de biblioteca con la opción LIBPATH_SUFFIX. Esta opción añade directorios al final de la vía de acceso a biblioteca, tras la vía de acceso a biblioteca base. Utilice esta opción para especificar directorios que contengan cualquier biblioteca nativa adicional que utilicen sus aplicaciones o cualquier servicio que no se incluya en la configuración de JVM estándar para CICS. Por ejemplo, las bibliotecas nativas adicionales pueden incluir los archivos DLL que son necesarios para utilizar los controladores JBDC de DB2.

La opción LIBPATH_PREFIX añade directorios al principio de la vía de acceso a biblioteca, antes de la vía de acceso a biblioteca base. Utilice esta opción con cuidado, ya que si los archivos DLL de los directorios especificados tienen el mismo nombre que los archivos DLL de la vía de acceso a biblioteca base, se cargan en lugar de los archivos proporcionados.

Todos los archivos DLL que se incluyan en la vía de acceso a biblioteca para que los utilicen sus aplicaciones se deben compilar y enlazar con la opción XPLink para un óptimo rendimiento. Los archivos DLL proporcionados en la vía de acceso a biblioteca base y los archivos DLL utilizados por servicios como los controladores DB2 JDBC se compilan con la opción XPLink.

OSGI_BUNDLES=*nombres_vía_de_acceso*
Especifica la vía de acceso del directorio para paquetes de middleware que se habiliten en la infraestructura OSGi de un servidor de JVM. Estos paquetes OSGi contienen clases para implementar funciones del sistema, como conectar a DB2 y a WebSphere MQ. Si especifica más de un paquete OSGi, utilice comas para separarlos.

Esta opción no se aplica a las JVM en agrupación.

OSGI_FRAMEWORK_TIMEOUT=60|*número*
Especifica cuántos segundos espera CICS a que la infraestructura OSGi se inicialice o se cierre antes de finalizar el tiempo de espera. Puede definir un valor de 1 a 60000 segundos. El valor predeterminado es 60 segundos. Si la infraestructura OSGi necesita más tiempo para el inicio que el número de segundos especificado, el servidor de JVM no puede inicializarse y CICS emite un mensaje DFHSJ0215. Los mensajes de error también se graban en los archivos de registro del servidor de JVM en zFS.Si la infraestructura necesita más tiempo para cerrarse que el número de segundos especificado, el servidor de JVM no se cerrará normalmente.

Esta opción no se aplica a las JVM en agrupación.

PRINT_JVM_OPTIONS={YES|NO}
Si esta opción se define como YES, siempre que se inicia una JVM todas las opciones que se pasan a dicha JVM en el inicio se imprimen en SYSPRINT. La salida se produce cada vez que se lanza una JVM con esta opción en su perfil. Puede utilizar esta opción para comprobar el contenido de las vías de acceso de clases para un perfil de JVM concreto, incluida la vía de acceso a biblioteca base y la vía de acceso de clases base creadas por CICS, que no son visibles en el perfil de JVM,.

REUSE={YES|NO}
Especifica si una JVM en agrupación es reutilizable o no:

- REUSE=YES, que es el valor predeterminado, crea una JVM que las aplicaciones Java reutilizan varias veces. Este tipo de JVM en agrupación se conoce como JVM continua.
- REUSE=NO crea una JVM que no se reutiliza, sino que se destruye tras ejecutarse en ella de un único programa Java. Este tipo de JVM en agrupación se conoce como JVM de un solo uso.

Esta opción no se aplica a los servidores de JVM.

STDERR={dfhjvmerr|*nombre_archivo***} [-generate]**
Especifica el nombre del archivo z/OS UNIX que utilizar para stderr. Si el archivo no existe, se crea en el directorio especificado por la opción WORK_DIR. Si el archivo existe, se añade la salida al final del archivo. Cuando la JVM se detiene, si el archivo stderr está vacío y se ha creado para la JVM específica, se suprime. En caso contrario, el archivo se conserva.

- En el caso de JVM en agrupación, el nombre predeterminado es `dfhjvmerr`. En el caso de un nombre de archivo fijo, la salida de varias JVM se añade al archivo indicado y la salida se intercala. Para crear archivos de salida exclusivos para cada JVM, utilice los símbolos &JVM_NUM; y &APPLID; en el nombre de archivo, como se enseña en los perfiles de JVM de ejemplo, o especifique la opción **-generate**. La opción **-generate** añade el número de JVM exclusivo, el APPLID de la región CICS e información identificativa adicional para el nombre de archivo. **-generate** debe ir precedida por un espacio en blanco.
- En el caso de servidores de JVM, el nombre de archivo es *applid.jvmserver.*`dfhjvmerr`. CICS crea automáticamente archivos de salida exclusivos para cada servidor de JVM mediante los símbolos &APPLID; y &JVMSERVER;.

Si se especifica la opción `USEROUTPUTCLASS` en un perfil de JVM, la clase Java indicada en dicha opción gestiona las solicitudes de System.err en su lugar. El archivo z/OS UNIX indicado por la opción `STDERR` todavía se puede utilizar si la clase indicada por la opción `USEROUTPUTCLASS` no puede grabar datos en su destino pretendido, como sucede si se utiliza la clase de ejemplo que se proporciona com.ibm.cics.samples.SJMergedStream. También puede utilizar el archivo si la salida la dirige a él por cualquier otra razón una clase indicada por la opción `USEROUTPUTCLASS`.

STDIN={dfhjvmin|*nombre_archivo***}**
Especifica el nombre del archivo z/OS UNIX que utilizar para `stdin`. Si el archivo no existe, se crea en el directorio especificado por la opción `WORK_DIR`.

STDOUT={dfhjvmout|*nombre_archivo***} [-generate]**
Especifica el nombre del archivo z/OS UNIX que utilizar para la salida del archivo `stdout`. Si el archivo no existe, se crea en el directorio especificado por la opción `WORK_DIR`. Si el archivo existe, se añade la salida al final del archivo. Cuando la JVM se detiene, si el archivo `stdout` está vacío y se ha generado para la JVM específica, se suprime. En caso contrario, el archivo se conserva.

- En el caso de JVM en agrupación, el nombre predeterminado es `dfhjvmout`. En el caso de un nombre de archivo fijo, la salida de varias JVM se añade al archivo indicado y la salida se intercala. Para crear archivos de salida exclusivos para cada JVM, utilice los símbolos &JVM_NUM; y &APPLID; en el nombre de archivo, como se enseña en los perfiles de JVM de ejemplo o especifique la opción **-generate**.
- En el caso de servidores de JVM, el nombre de archivo es *applid.jvmserver.*`dfhjvmout`. CICS crea automáticamente archivos de salida exclusivos para cada servidor de JVM mediante los símbolos &APPLID; y &JVMSERVER;.

Si se especifica la opción `USEROUTPUTCLASS` en un perfil de JVM, la clase Java indicada en dicha opción gestiona las solicitudes de System.out en su lugar. El archivo z/OS UNIX indicado por la opción `STDOUT` todavía se puede utilizar si la clase indicada por la opción `USEROUTPUTCLASS` no puede grabar datos en su destino pretendido, por ejemplo, si se utiliza la clase de ejemplo com.ibm.cics.samples.SJMergedStream. También puede utilizar el archivo si la salida la dirige a él por cualquier otra razón una clase indicada por la opción `USEROUTPUTCLASS`.

USEROUTPUTCLASS={*nombre_clase***}**
Especifica el nombre completo de una clase Java que intercepta la salida de la JVM y mensajes de los componentes internos de la JVM. Puede utilizar esta clase Java para redirigir la salida y los mensajes de las JVM, y puede añadir indicaciones de fecha y hora y cabeceras a los registros de salida. Si la clase

Java no puede grabar datos en su destino pretendido, los archivos indicados en las opciones STDOUT y STDERR todavía se pueden utilizar.

Especificar la opción USEROUTPUTCLASS tiene un efecto negativo sobre el rendimiento de las JVM. Para lograr el mejor rendimiento en un entorno de producción, no utilice esta opción. Sin embargo, esta opción puede resultar útil para los desarrolladores de aplicaciones que utilicen la misma región CICS, ya que la salida de la JVM se puede dirigir a un destino identificable.

Para obtener más información sobre esta clase y los ejemplos provistos, consulte "Control de la ubicación para las salidas de JVM stdout, stderr y de volcado" en la página 197.

WORK_DIR={.|*nombre_directorio***}**
Especifica el directorio de trabajo en z/OS UNIX que la región CICS utiliza para actividades relacionadas con Java. La interfaz de JVM de CICS utiliza este directorio al crear los archivos stdin, stdout y stderr. Se define un punto (.) en los perfiles de JVM provistos, lo que indica que el directorio de inicio del ID de usuario de la región CICS (es decir, el directorio /u/*id_usuario_región_CICS* de z/OS UNIX) se va a utilizar como directorio de trabajo. Este directorio se puede crear durante la instalación de CICS. Si el ID de usuario de la región CICS no tiene este directorio de inicio, o si se omite WORK_DIR, se utiliza /tmp como nombre de directorio de z/OS UNIX.

Subdirectorio de trabajo relativo
Solo en el caso de que se trate de JVM agrupados, podrá crear un subdirectorio relativo en este directorio z/OS UNIX para mantener los archivos de salida, especificando el nombre del subdirectorio después del punto. Por ejemplo, si especifica:

```
WORK_DIR=./javaoutput
```

los archivos de salida de todas las JVM en esa región CICS se crean en el subdirectorio javaoutput del directorio de inicio del ID de usuario de la región CICS.

Directorio de trabajo absoluto
Cuando se trate de JVM agrupados y servidores JVM, podrá especificar una vía de acceso absoluta para el directorio de trabajo. Si no desea utilizar el directorio de inicio como el directorio de trabajo para actividades relacionadas con Java, o si sus regiones CICS comparten el mismo ID de usuario (UID) de z/OS y, por lo tanto, tienen el mismo directorio de inicio, puede crear un directorio de trabajo distinto para cada región CICS. Se especifica un nombre de directorio que utiliza el símbolo &APPLID;, para lo que CICS sustituye el identificador de aplicación real de la región CICS. Por lo que puede tener un directorio de trabajo exclusivo para cada región, incluso si todas las regiones CICS comparten el mismo conjunto de perfiles de JVM. Por ejemplo, si especifica:

```
WORK_DIR=/u/&APPLID;/javaoutput
```

cada región CICS que utiliza ese perfil de JVM tiene su propio directorio de trabajo. Asegúrese de haber creado todos los directorios relevantes en z/OS UNIX y de haberles dado a las regiones CICS acceso de lectura, grabación y ejecución a ellos.

También puede especificar un nombre fijo para el directorio de trabajo, de nuevo asegurándose de haber creado el directorio correspondiente en z/OS UNIX y de haber otorgado a las regiones CICS el acceso

correcto. Cuando se utiliza un nombre fijo para el directorio de trabajo, los archivos de salida de todas las JVM de las regiones CICS que comparten el perfil de JVM se crean en ese directorio. Si también ha utilizado nombres de archivo fijos para los archivos de salida, la salida de todas las JVM de estas regiones CICS se añade a los mismos archivos z/OS UNIX. Para evitar añadir a los mismos archivos, utilice el símbolo &JVM_NUM; y los símbolos &APPLID; con las opciones de perfil de JVM apropiadas para producir archivos de volcado y de salida únicos para cada JVM de cada región de CICS.

No defina los directorios de trabajo en el directorio CICS de z/OS UNIX, que es el directorio de inicio para archivos de CICS según lo definido por el parámetro de inicialización del sistema **USSHOME**.

También puede utilizar la opción USEROUTPUTCLASS para indicar una clase Java que intercepte, redirija y formatee la salida stderr y stdout desde una JVM. Las clases de ejemplo provistas para redirección de salida utilizan el directorio especificado por WORK_DIR en algunas circunstancias.

-generate
Especifique esta opción para identificar de forma exclusiva los archivos stdout (salida de JVM) y stderr (mensajes de error de JVM) en z/OS UNIX. Esta opción se debe especificar en las opciones STDOUT y STDERR tras el nombre de archivo. Por ejemplo, puede especificar STDOUT=dfhjvmout -generate.

La opción **-generate** añade al número de JVM exclusivo (al igual que el símbolo &JVM_NUM;), el ID de aplicación de la región CICS (como el símbolo &APPLID;) y algunos calificadores adicionales al nombre de archivo que se haya especificado para la opción STDOUT o STDERR.

Por ejemplo, un archivo stdout típico creado con la opción **-generate** puede tener el siguiente nombre:

```
dfhjvmout.IYK2ZIK1.0067240142.06004165342.txt
```

donde:

- dfhjvmout es la parte fija del nombre de archivo.
- IYK2ZIK1 es el ID de aplicación de la región CICS.
- 0067240142 es el número de JVM exclusivo.
- 06004165342 es la indicación de fecha y hora que muestra cuándo se creó la JVM.
- .txt es el sufijo de archivo.

Cuando se utiliza la opción **-generate**, los símbolos &APPLID; y &JVM_NUM; no son necesarios en el nombre de archivo, ya que **-generate** proporciona estos fragmentos de información de forma automática.

Como la opción **-generate** incluye el número de JVM, el archivo de salida resultante es exclusivo para la JVM y se puede hacer coincidir con el número de JVM identificado desde el mandato **INQUIRE JVM**. Como incluye el identificador de aplicación de la región CICS, también es exclusivo en varias regiones CICS.

Esta opción no se aplica a los servidores de JVM.

-Xdebug
Especifica si hay que habilitar el soporte de depuración en la JVM.

Para obtener más información, consulte el apartado "Depuración de una aplicación Java" en la página 206. Consulte también la información relativa a Java Platform Debugger Architecture (JPDA) disponible en el sitio web Oracle Technology Network Java.

Para asegurarse de que la sesión de depuración para JVM en agrupación tenga un final limpio, especifique REUSE=NO cuando se habilite el soporte de depuración.

-Xms
Especifica el tamaño inicial del almacenamiento dinámico. Especifique los tamaños de almacenamiento en múltiplos de 1024 bytes. Utilice la letra K para indicar KB, la letra M para indicar MB y la letra G para indicar GB. Por ejemplo, para especificar 6.291.456 bytes como el tamaño inicial del almacenamiento dinámico, codifique **-Xms** de uno de estos modos:

```
-Xms6144K
-Xms6M
```

Especifique *tamaño* como un número de KB o MB. Para obtener información sobre el valor predeterminado, consulte Default settings for the JVM.

-Xmx
Especifica el tamaño máximo del almacenamiento dinámico. Esta cantidad fija de almacenamiento la asigna la JVM durante la inicialización de la JVM.

Especifique *tamaño* como un número de KB o MB.

-Xshareclasses
Especifique esta opción para habilitar el uso compartido de datos de clase en una memoria caché de clase compartida. La JVM se conecta a una memoria caché existente o crea una memoria caché si no existe ninguna. Puede tener varias memorias caché y puede especificar la memoria caché correcta añadiendo una subopción para la opción **-Xshareclasses**. Para obtener más detalles, consulte Class data sharing between JVMs.

Esta opción no se aplica a las JVM en agrupación.

Propiedades del sistema de la JVM

Las propiedades del sistema contienen información para configurar la máquina virtual Java y su entorno. Algunas propiedades del sistema son especialmente relevantes para JVM en un entorno de CICS.

Especifique propiedades del sistema de JVM en el perfil de JVM. En el caso de JVM en agrupación, también puede definir estas opciones en un archivo de propiedades de JVM para que distintos perfiles compartan las mismas opciones. Para hacer referencia al archivo, utilice la opción JVMPROPS en cada perfil de JVM. CICS pasa todas las propiedades del sistema en un perfil de JVM o en un archivo de propiedades de JVM a la JVM sin modificar.

Si utiliza archivos de propiedades de JVM, asegúrese de que los archivos sean seguros, con la autorización de actualización restringida a los administradores del sistema, si se utilizan para definir opciones de configuración de JVM confidenciales; por ejemplo, el archivo de política de seguridad.

La JVM puede soportar un rango de propiedades del sistema mucho mayor que las documentadas aquí. "Perfiles de JVM: opciones y ejemplos" en la página 104 indica algunas fuentes de información recomendadas sobre propiedades del sistema.

La siguiente lista incluye una selección de propiedades del sistema relevantes y describe cómo puede utilizarlas en un entorno de CICS. Las propiedades del sistema que comienzan con **-Dcom.ibm.cics** son específicas para la JVM IBM en un entorno de CICS. Las que comienzan por **-Dcom.ibm** (sin **.cics**) o por **-Djava** se utilizan más habitualmente. Especifique cada propiedad del sistema con arreglo a las reglas de codificación descritas en "Reglas para codificación de perfiles de JVM" en la página 106.

-Dcom.ibm.cics.datasource.path=
Especifica el nombre y el subcontexto de un DataSource compatible con CICS que haya desplegado para generar conexiones de JDBC para aplicaciones Java en CICS que accedan a DB2. Para obtener más información, consulte Adquisición de una conexión mediante una interfaz DataSource en la Guía DB2.

-Dcom.ibm.cics.ejs.nameserver=
Especifica el URL y el número de puerto TCP/IP del servidor de nombres que se utiliza para referencias de JNDI.

- En el caso de un servidor de nombres LDAP, especifique algo similar a esto:

  ```
  -Dcom.ibm.cics.ejs.nameserver=ldap://myldserv.example.com:389
  ```

  `myldserv.example.com` es el URL del servidor de nombres y `389` es el número de puerto en el que está configurado para escuchar. Su administrador de LDAP puede proporcionar el URL y el número de puerto correctos.
- En el caso de un CORBA Object Services Naming Directory Server, especifique algo similar a:

  ```
  -Dcom.ibm.cics.ejs.nameserver=iiop://mycsserv.example.com:900
  ```

  El administrador correspondiente de su organización puede proporcionar el nombre y el número de puerto correctos.

  Si utiliza el CORBA Object Services Naming Directory Server proporcionado con WebSphere Application Server, especifique algo similar a:

```
-Dcom.ibm.cics.ejs.nameserver=iiop://mycsserv.example.com:2809/domain/legacyRoot
```

  En WebSphere Application Server, se aplican estas condiciones:
  - El puerto TCP/IP predeterminado utilizado por el CORBA Object Services Naming Directory Server es el 2809.
  - Los objetos de CICS se deben publicar en una ubicación diseñada específicamente en la estructura de denominación de WebSphere llamada `domain/legacyRoot`. CICS publica objetos en un contexto definido por la opción JNDIPREFIX de la definición CORBASERVER, donde el prefijo de JNDI es una vía de acceso relativa. Si no especifica la vía de acceso `/domain/legacyRoot` desde el nodo raíz del espacio de nombres, CICS intenta publicar objetos en la ubicación del prefijo de JNDI relativa al propio nodo raíz. Con el CORBA Object Services Naming Directory Server proporcionado con WebSphere Application Server, este intento no tiene éxito.

Si utiliza un servicio de denominación CORBA Object Services y ha elegido especificarlo en **-Djava.naming.provider.url**, no lo vuelva a especificar aquí.

-Dcom.ibm.cics.ejs.loadjndiproperties=
Configura un archivo llamado `jndi.properties` para que contenga propiedades de configuración del servidor de nombres de JNDI que son comunes en un conjunto de regiones CICS. De forma predeterminada, CICS no intenta ubicar

un archivo `jndi.properties`. Incluya la siguiente propiedad del sistema para hacer que CICS cargue `jndi.properties` para esta JVM:

```
-Dcom.ibm.cics.ejs.loadjndiproperties=true
```

Coloque el directorio que contenga el archivo `jndi.properties` en la vía de acceso de clase estándar del perfil de JVM, en todos los perfiles de JVM relevantes, para todas las regiones que desee que compartan los mismos valores de servidor de nombres.

-Dcom.ibm.cics.iiop.CSIv2Enabled=true
Habilita el soporte de CICS para el protocolo Common Secure Interoperability versión 2 (CSIv2) para aserción de identidad. Para activar este soporte, especifique esta propiedad del sistema en todos los perfiles de JVM o en los archivos de propiedades de JVM utilizados en la región CICS. Este soporte es necesario si un CorbaServer de CICS debe soportar autenticación de identidad con aserción para mensajes de IIOP enviados desde WebSphere Application Server for z/OS.

-Dcom.ibm.cics.soap.validation.local.CCSID=
Especifica la página de códigos local que utilizar al validar mensajes SOAP si la validación está habilitada para un recurso de CICS WEBSERVICE. Si no se especifica ningún CCSID local, se asume la página de códigos USS predeterminada para la instalación al validar el mensaje SOAP.

-Dcom.ibm.websphere.naming.jndicache.cacheobject={populated |none}
Activa y desactiva la memoria caché de JNDI. La memoria caché de JNDI almacena los resultados de las búsquedas JNDI en almacenamiento local, de forma que, si una aplicación hace la misma búsqueda dos veces, quizá en tareas distintas, los resultados ya estén disponibles. La memoria caché tiene estas características:

- Es específica de JVM. Es decir, existe una memoria caché distinta para cada JVM.
- Trabaja solo con un servidor de nombres IBM JNDI.
- Almacena únicamente referencias de objeto y no otros elementos, como DataSources.

populated
La memoria caché JNDI está activa.

none La memoria caché JNDI no se utiliza.

-Dcom.ibm.websphere.naming.jndicache.maxcachelife={20 *vmins***}**
Especifica, en minutos, la "vida" de la memoria caché JNDI. Si se accede a la memoria caché tras superarse este tiempo, se vacía todo el contenido de dicha memoria.

Consulte también la propiedad **-Dcom.ibm.websphere.naming.jndicache.cacheobject**.

-Dcom.ibm.ws.naming.ldap.containerdn=
Especifica el nombre distinguido de contenedor para el servidor de nombres LDAP. Por ejemplo:

```
-Dcom.ibm.ws.naming.ldap.containerdn=ibm-wsnTree=t1,o=WASNaming,c=us
```

Su administrador de LDAP puede proporcionar el valor correcto para la instalación.

El nombre distinguido de contenedor es la raíz del espacio de nombres del sistema.

Esta propiedad no es necesaria si se especifica un servicio de denominación de COS.

-Dcom.ibm.ws.naming.ldap.noderootrdn=
Especifica el nombre distinguido relativo del nodo raíz para el servidor de nombres LDAP. Por ejemplo:

```
-Dcom.ibm.ws.naming.ldap.noderootrdn=ibm-wsnName=legacyroot,
ibm-wsnName=PLEX2,ibm-wsnName=domainRoots
```

Su administrador de LDAP puede proporcionar el valor correcto.

Esta propiedad no es necesaria si se especifica un servicio de denominación COS.

-Dfile.encoding=
Especifica la codificación.

-Djava.naming.security.authentication=
Especifica el tipo de autenticación de seguridad utilizado para operaciones de denominación. Tal vez necesite esta propiedad si utiliza un servidor de nombres LDAP.

CICS debe tener acceso de grabación al espacio de nombres LDAP. Si el servicio LDAP se configura de forma segura, son necesarias estas tres propiedades: autenticación, credenciales y principal. Si el servicio LDAP se configura de forma que cualquier usuario pueda grabar en él, no son necesarias estas tres propiedades. El administrador de LDAP puede indicarle si es necesario incluir estas propiedades en el perfil de JVM o en un archivo de propiedades de JVM opcional.

El único valor de esta propiedad que admite CICS es **simple**. Especificar **-Djava.naming.security.authentication=simple** indica que el servidor de nombres LDAP se está ejecutando en modalidad segura.

Importante:

Si se especifica esta propiedad, también debe especificar **-Djava.naming.security.principal** y **-Djava.naming.security.credentials**.

Dado que estas propiedades contienen el ID de usuario y la contraseña que CICS necesita para acceder al servicio LDAP seguro, debe prestar especial atención a los permisos de archivo que controlan el acceso a todos los archivos que contienen estas propiedades del sistema.

-Djava.naming.security.credentials=
Especifica la contraseña necesaria para **principal**, que se describe en java.naming.security.principal, para acceder al servidor de nombres LDAP.

Esta propiedad es necesaria si se ha especificado **-Djava.naming.security.authentication=simple**. Su administrador de LDAP proporciona el valor que hay que especificar. Por ejemplo:

```
-Djava.naming.security.credentials=secret
```

.

-Djava.naming.security.principal=
Especifica el **principal** necesario para acceder al servidor de nombres LDAP.

Esta propiedad es necesaria si se ha especificado **-Djava.naming.security.authentication=simple**. Su administrador de LDAP

proporciona el valor que hay que especificar. Por ejemplo, **-Djava.naming.security.principal=cn=CICSUser,c=uk** .

-Djava.security.manager={default| "" | |*otro_gestor_seguridad*}
Especifica el gestor de seguridad Java que habilitar para la JVM. Para habilitar el gestor de seguridad Java predeterminado, incluya esta propiedad del sistema en uno de los siguientes formatos:

- `-Djava.security.manager=default`
- `-Djava.security.manager=""`
- `-Djava.security.manager=`

Todas estas sentencias habilitan el gestor de seguridad predeterminado. Si no incluye la propiedad del sistema **-Djava.security.manager** en el perfil de JVM, la JVM se ejecuta sin la seguridad Java habilitada. Para inhabilitar la seguridad Java para una JVM, elimine esta propiedad del sistema.

-Djava.security.policy=
Describe la ubicación de archivos de política adicionales que se desea que utilice el gestor de seguridad para determinar la política de seguridad para la JVM. Se proporciona un archivo de política predeterminado con la JVM en `/usr/lpp/java/J6.0.1_64/lib/security/java.policy`, donde los nombres del subdirectorio `java/J6.0.1_64` son los valores predeterminados cuando se instala el IBM 64 bits SDK para z/OS, Java Technology Edition. El gestor de seguridad predeterminado siempre utiliza este archivo de política predeterminado para determinar la política de seguridad para la JVM, y se puede utilizar la propiedad del sistema **-Djava.security.policy** para especificar cualquier archivo de política adicional que desee que el gestor de seguridad tenga en cuenta además del archivo de política predeterminado.

Para permitir que las aplicaciones Java de CICS se ejecuten correctamente cuando la seguridad Java está activa, especifique, como mínimo, un archivo de política adicional que brinde a CICS los permisos necesarios para ejecutar la aplicación.

Para obtener información sobre cómo habilitar la seguridad Java, consulte "Habilitación del gestor de seguridad Java" en la página 93.

-Djdbc.drivers=
Especifica uno o más controladores JBDC de 64 bits. Definir los nombres de controlador como una propiedad del sistema es una alternativa a que la propia aplicación Java cargue los controladores utilizando el mandato **Class.forName("nombre_controlador");**. Separe cada nombre de controlador en una lista mediante el símbolo : (dos puntos).

Para especificar los controladores JDBC incluidos en DB2, defina la propiedad del sistema:

```
-Djdbc.drivers=com.ibm.db2os390.sqlj.jdbc.DB2SQLJDriver
```

Este nombre común funciona para todos los niveles del controlador JDBC proporcionado por DB2, incluido el controlador DB2 Universal JDBC Driver.

Debe utilizar la versión de 64 bits de los controladores JBDC. Para obtener más información sobre JDBC, consulte Uso de JDBC y SQLJ para acceder a datos de DB2 desde programas Java en la Guía DB2.

DFHJVMAX, perfil de JVM para el servidor de JVM

El perfil de JVM DFHJVMAX es un perfil de JVM proporcionado por CICS que es utilizado por un servidor de JVM Axis2. Asegúrese de que DFHJVMAX se configura correctamente para su región CICS.

Opciones de JVM en el perfil de JVM DFHJVMAX

```
########################################################################
# Perfil de JVM: DFHJVMAX                                              #
#                                                                      #
# Este perfil JVM de CICS de ejemplo es para un servidor JVM Axis2.          #
#                                                                      #
########################################################################
#
#                         Sustitución de símbolo
#                         ----------------------
#
# Las siguientes sustituciones está soportadas:
#   &APPLID;     => El ID de aplicación de la región CICS.
#   &JVMSERVER;  => El nombre del recurso JVMSERVE.
#   &DATE;       => Fecha en la que se ha habilitado JVMSERVER. Daammdd
#   &TIME;       => Hora a la que se ha habilitado JVMSERVER. Thhmmss
#
# Utilizar sustituciones significa que se puede utilizar el mismo perfil
# para varias regiones y seguir teniendo directorios de trabajo y destinos
# de salida exclusivos para cada región. #
# Con esta sustitución
#     ENV_VAR=myvar.&APPLID;.&JVMSERVER;.data
# se convierte en
#     ENV_VAR=myvar.ABCDEF.JSERVER1.data
# para un recurso JVMSERVER con el nombre JSERVER1 en una región CICS con el
# ID de aplicación ABCDEF. #
#***********************************************************************
#
#                         Parámetros necesarios
#                         ---------------------
#
# El conjunto de opciones de CICS admitidas para servidores de JVM
# difiere de las utilizadas con agrupaciones de  JVM. #
# JAVA_HOME especifica la ubicación del directorio de Java. #
JAVA_HOME=/usr/lpp///&JAVA_HOME///
#
# Defina el directorio de trabajo actual. Si esta variable de entorno se
# define, se emite un cambio al directorio especificado antes de que la JVM
# se inicialice, y las secuencias STDIN, STDOUT y STDERR se
# asignan a este directorio. #
# Si no se especifica esta opción, el directorio de trabajo actual se
# deja sin cambiar y las secuencias STDIN, STDOUT y STDERR se asignan
# al directorio /tmp.
#
WORK_DIR=.
#
# Especifique los directorios que contienen las DLL necesarias en tiempo de ejecución.
# Por ejemplo, para utilizar el controlador de DB2 de IBM para JDBC y SQLJ, añada el
# directorio que contenga las DLL nativas a la opción LIBPATH_SUFFIX.
# Consulte la "Application Programming Guide and Reference for Java" aplicable
# al nivel de DB2 que se utiliza. #
#LIBPATH_SUFFIX=
#
#***********************************************************************
#
#                     Parámetros específicos de servidor de JVM
#                     -----------------------------
#
# Utilice la opción JAVA_PIPELINE para configurar el servidor de JVM
# para que soporte interconexiones SOAP basadas en Java. #
JAVA_PIPELINE=YES
#
#***********************************************************************
#
#                         Redirección de salida
#                         ---------------------
#
```

```
# STDOUT, STDERR, STDIN y JVMTRACE se asignan con nombres de archivo
# que comiencen por &APPLID;.&JVMSERVER;. Puede especificar
# nombres de archivo distintos mediante las variables de entorno STDOUT, STDERR, STDIN y
# JVMTRACE. #
# El nombre de archivo predeterminado para JVMTRACE es dfhjvmtrc.
# Para enviar la salida a algún lugar distinto a un archivo, especifique una clase
# de redirección de salida de usuario. CICS proporciona un ejemplo que enseña
# esta prestación. JVMTRACE no se puede redirigir. #
#USEROUTPUTCLASS=com.ibm.cics.samples.SJMergedStream
#
#*********************************************************************
#
#                              Opciones de JVM
#                              -----------
#
# Consulte "IBM SDK for z/OS platforms, Java Technology Edition, SDK Guide"
# o "IBM Developer Kit and Runtime Environment, Java Technology
# Edition, Diagnostics Guide" para obtener información sobre todas las opciones de JVM. #
# Las opciones de JVM que imprimen salida y luego salen no se deben especificar,
# ya que harán que no se produzca la creación de la JVM. Estas
# opciones incluyen: -version, -help, -?, -assert y -X. #
# Utilice las siguientes opciones para ajustar la JVM.
# -Xms    Tamaño de almacenamiento dinámico de Java inicial; por ejemplo, -Xms64M
# -Xmx    Tamaño de almacenamiento dinámico de Java máximo; por ejemplo, -Xmx512M
# -Xmso   Tamaño de pila inicial para hebras nativas (valor predeterminado -Xmso256KB)
# -Xiss   Tamaño de pila inicial para hebras Java (valor predeterminado -Xiss128KB)
# -Xss    Tamaño de pila máximo para hebras Java (valor predeterminado -Xss256KB)
#
#
# Omita estos valores para aceptar los valores predeterminados de la JVM,
# a no ser que haya realizado un análisis de la carga de trabajo y pueda
# proporcionar valores ajustados para una carga de trabajo estable.
#
# La opción -Xgcthreads define el número máximo de hebras de ayudante
# permitidas para recogida de basura. Si no se especifica esta opción,
# el valor predeterminado se define como (número de CPUs - 1).
#
# -Xgcthreads4
#
# La siguiente opción define la política de recogida de basura.
#
-Xgcpolicy:gencon#
#*********************************************************************
#
#                 Definición de propiedades del sistema de JVM del usuario
#                 --------------------------------
#
# Especifique las propiedades del sistema de JVM para un servidor de JVM
# si es necesario.
# Las propiedades son pares de nombre de clave y valor que
# contienen información básica sobre la JVM y su entorno. Siempre
# llevan el prefijo -D. Por ejemplo:  #
# -Dcom.ibm.cics.some.property=algún_valor
#
#*********************************************************************
#
#              Variables de entorno de UNIX System Services
#              ----------------------------------------
#
# Opciones de volcado de Java. Consulte "IBM Developer Kit and Runtime Environment,
# Java Technology Edition Diagnostics Guide" o "IBM SDK for z/OS
# platforms, Java Technology Edition, SDK Guide" para obtener información sobre
# todas las opciones de tiempo de ejecución de Java. #
JAVA_DUMP_OPTS="ONANYSIGNAL(JAVADUMP,SYSDUMP),ONINTERRUPT(NONE)"
#
# Especifique dónde se graban los volcados de la JVM
#
```

```
#JAVA_DUMP_TDUMP_PATTERN=DUMP.&APPLID;.&JVMSERVER;.&DATE;.&TIME;
#
# Especifique el huso horario local
#
#TZ=CET-1CEST,M3.5.0,M10.5.0
#
```

DFHOSGI, perfil de JVM para el servidor de JVM

El perfil de JVM DFHOSGI es un perfil de JVM proporcionado por CICS que es utilizado por un servidor de JVM Axis2. Asegúrese de que DFHOSGI se configura correctamente para su región CICS.

Opciones de JVM en el perfil de JVM DFHJVMCD

```
########################################################################
# Perfil de JVM: DFHOSGI                                               #
#                                                                      #
# Este perfil JVM de CICS de ejemplo es para un servidor JVM.                  #
#                                                                      #
########################################################################
#
#                          Sustitución de símbolo
#                          ----------------------
#
# Las siguientes sustituciones está soportadas:
#   &APPLID;     => El ID de aplicación de la región CICS.
#   &JVMSERVER;  => El nombre del recurso JVMSERVE.
#   &DATE;       => Fecha en la que se ha habilitado JVMSERVER. Daammdd
#   &TIME;       => Hora a la que se ha habilitado JVMSERVER. Thhmmss
#
# Utilizar sustituciones significa que se puede utilizar el mismo perfil
# para varias regiones y seguir teniendo directorios de trabajo y destinos
# de salida exclusivos para cada región. #
# Con esta sustitución
#     ENV_VAR=myvar.&APPLID;.&JVMSERVER;.data
# se convierte en
#     ENV_VAR=myvar.ABCDEF.JSERVER1.data
# para un recurso JVMSERVER con el nombre JSERVER1 en una región CICS con el
# ID de aplicación ABCDEF. #
# Nota: el carácter de continuación para su uso con perfiles de JVM es '\'.
#**********************************************************************
#
#                          Parámetros necesarios
#                          ---------------------
#
# El conjunto de opciones de CICS admitidas para servidores de JVM
# difiere de las utilizadas con agrupaciones de  JVM. #
# JAVA_HOME especifica la ubicación del directorio de Java. #
JAVA_HOME=/usr/lpp///&JAVA_HOME///
#
# Defina el directorio de trabajo actual. Si esta variable de entorno se
# define, se emite un cambio al directorio especificado antes de que la JVM
# se inicialice, y las secuencias STDIN, STDOUT y STDERR se
# asignan a este directorio. #
# Si no se especifica esta opción, el directorio de trabajo actual se
# deja sin cambiar y las secuencias STDIN, STDOUT y STDERR se asignan
# al directorio /tmp.
#
WORK_DIR=.

# Especifique los directorios que contienen las DLL necesarias en tiempo de ejecución.
# Por ejemplo, para utilizar el controlador de DB2 de IBM para JDBC y SQLJ, añada el
# directorio que contenga las DLL nativas a la opción LIBPATH_SUFFIX.
# Consulte la "Application Programming Guide and Reference for Java" aplicable
# al nivel de DB2 que se utiliza. #
#LIBPATH_SUFFIX=
```

```
#
#*********************************************************************
#
#                   Parámetros específicos de servidor de JVM
#                   -----------------------------
#
# La opción OSGI_BUNDLES se utiliza para especificar una lista de paquetes
# de middleware que se instalan y activan en la infraestructura OSGi
# al inicializar la JVM.
# La lista de paquetes debe estar separada por comas. El carácter de
# continuación es '\'.
#
#OSGI_BUNDLES=/u/example/pathToBundleDirectory/B1.jar,\
#             /u/example/pathToBundleDirectory/B2.jar
#
#Esta opción se utiliza para especificar, en segundos, cuánto se permite
# ejecutar la inicialización de infraestructura, la terminación y los
# paquetes de middleware antes de finalizar el tiempo de espera.
# El valor especificado debe estar entre 1 y 60000. Si está fuera
# de este rango, adoptará por defecto el valor 60. Si la
# inicialización excede el límite, el servidor de JVM no se inicializará.
#
#OSGI_FRAMEWORK_TIMEOUT=60
#
#*********************************************************************
#
#                         Redirección de salida
#                         ------------------
#
# STDOUT, STDERR, STDIN y JVMTRACE se asignan con nombres de archivo
# que comiencen por &APPLID;.&JVMSERVER;. Puede especificar
# nombres de archivo distintos mediante las variables de entorno STDOUT, STDERR,
# STDIN y JVMTRACE. #
# El nombre de archivo predeterminado para JVMTRACE es dfhjvmtrc.
# Para enviar la salida a algún lugar distinto a un archivo, especifique una clase
# de redirección de salida de usuario. CICS proporciona un ejemplo que enseña
# esta prestación. JVMTRACE no se puede redirigir. #
#USEROUTPUTCLASS=com.ibm.cics.samples.SJMergedStream
#
#*********************************************************************
#
#                             Opciones de JVM
#                             -----------
#
# Consulte "IBM SDK for z/OS platforms, Java Technology Edition, SDK Guide"
# o "IBM Developer Kit and Runtime Environment, Java Technology
# Edition, Diagnostics Guide" para obtener información sobre todas las opciones de JVM. #
# Las opciones de JVM que imprimen salida y luego salen no se deben especificar,
# ya que harán que no se produzca la creación de la JVM. Estas
# opciones incluyen: -version, -help, -?, -assert y -X. #
# Utilice las siguientes opciones para ajustar la JVM.
# -Xms    Tamaño de almacenamiento dinámico de Java inicial; por ejemplo, -Xms64M
# -Xmx    Tamaño de almacenamiento dinámico de Java máximo; por ejemplo, -Xmx512M
# -Xmso   Tamaño de pila inicial para hebras nativas (valor predeterminado -Xmso256KB)
# -Xiss   Tamaño de pila inicial para hebras Java (valor predeterminado -Xiss128KB)
# -Xss    Tamaño de pila máximo para hebras Java (valor predeterminado -Xss256KB)
#
# Omita estos valores para aceptar los valores predeterminados de la JVM,
# a no ser que haya realizado un análisis de la carga de trabajo y pueda
# proporcionar valores ajustados para una carga de trabajo estable.
#
# La opción -Xgcthreads define el número máximo de hebras de ayudante
# permitidas para recogida de basura. Si no se especifica esta opción,
# el valor predeterminado se define como (número de CPUs - 1).
#
# -Xgcthreads4
#
```

```
# La siguiente opción define la política de recogida de basura.
#
-Xgcpolicy:gencon#
#*********************************************************************
#
#                 Definición de propiedades del sistema de JVM del usuario
#                 --------------------------------
#
# Especifique las propiedades del sistema de JVM para un servidor de JVM
# si es necesario.
# Las propiedades son pares de nombre de clave y valor que
# contienen información básica sobre la JVM y su entorno. Siempre
# llevan el prefijo -D. Por ejemplo:  #
# -Dcom.ibm.cics.some.property=algún_valor
#
#*********************************************************************
#
#              Variables de entorno de UNIX System Services
#              ----------------------------------------
#
# Opciones de volcado de Java. Consulte "IBM Developer Kit and Runtime Environment,
# Java Technology Edition Diagnostics Guide" o "IBM SDK for z/OS
# platforms, Java Technology Edition, SDK Guide" para obtener información sobre
# todas las opciones de tiempo de ejecución de Java.
#
JAVA_DUMP_OPTS="ONANYSIGNAL(JAVADUMP,SYSDUMP),ONINTERRUPT(NONE)"
#
# Especifique dónde se graban los volcados de la JVM
#
#JAVA_DUMP_TDUMP_PATTERN=DUMP.&APPLID;.&JVMSERVER;.&DATE;.&TIME;
#
# Especifique el huso horario local
#
#TZ=CET-1CEST,M3.5.0,M10.5.0
#
```

DFHJVMPR, perfil de JVM para una JVM en agrupación

El perfil de JVM DFHJVMPR es un perfil de JVM de ejemplo para JVM en agrupación que utilizan la memoria caché de clase compartida. El archivo se utiliza como el predeterminado si no se especifica un perfil de JVM o un servidor de JVM en el recurso PROGRAM para el programa Java.

Opciones de JVM en el perfil de JVM DFHJVMPR

```
############################################################
# Perfil de JVM: DFHJVMPR
############################################################
#
# Este es un perfil de JVM de CICS de ejemplo para las JVM que utilizan
# la memoria caché de clase compartida. Este perfil es el perfil predeterminado
# para utilizar con todos los recursos PROGRAM de CICS definidos con JVM(YES),
# a no ser que se especifique algo distinto.
#
######
#
# Sustitución de símbolo:
#
# Si utiliza cualquiera de los siguientes símbolos variables en cualquiera de
# las variables de debajo, se sustituirá por los valores
# adecuados. Los símbolos variables pueden especificarse en mayúscula o
# minúscula.
#
#  Símbolo       Valor de sustitución
#  ------       -----------------
#
```

```
# &APPLID;      El APPLID de la región CICS.
# &JVM_NUM;     El ID de proceso (pid) de UNIX System Services
#               de la JVM. Está garantizado que es exclusivo.
# &DATE;        La fecha actual con el formato Daammdd.
# &TIME;        La hora actual con el formato Thhmmss
#
# Con esta sustitución, por ejemplo:
#     STDERR=dfhjvmerr.&APPLID;.&JVM_NUM;.&DATE;.&TIME;
# se convierte en
#     STDERR=dfhjvmerr.ABCDEF.0084214386.D081220.T135323
#
######
#
# ********* Parámetros específicos de CICS ***********
#
JAVA_HOME=/usr/lpp/java/J6.0.1_64
WORK_DIR=.
REUSE=YES
CLASSCACHE=YES
#
# Opcionalmente, se puede utilizar un archivo de propiedades de JVM especificando su
# vía de acceso y su nombre de archivo completos en la opción JVMPROPS.
# Consulte "Aplicaciones Java en CICS" para obtener más información sobre los archivos
# de propiedades de JVM.
#
# JVMPROPS=/u/example/pathToProperties/myJVMProps.data
#
STDIN=dfhjvmin
STDOUT=dfhjvmout
STDERR=dfhjvmerr
#
DISPLAY_JAVA_VERSION=NO
# Porcentaje de llenado del almacenamiento dinámico que desencadenará una recogida de
basura planificada.
GC_HEAP_THRESHOLD=85
# Valor de tiempo de espera en minutos tras el que una JVM y su TCB pasan a ser
# elegibles para terminación.
IDLE_TIMEOUT=30
#
# Especifique los directorios que contienen DLL necesarias en tiempo de ejecución.
# Por ejemplo, para utilizar el controlador DB2 de IBM para JDBC y SQLJ,
# añada el directorio que contenga las DLL nativas a
# LIBPATH_SUFFIX. Consulte la DB2 Application and Programming Guide.
# Guide for Java correspondiente al nivel de DB2 que se utiliza.
#
#LIBPATH_PREFIX=
#LIBPATH_SUFFIX=
#
# Especifique los directorios que contengan clases Java de aplicación
# y archivos JAR. (Elimine el comentario de las líneas de debajo si es necesario).
#
#CLASSPATH_SUFFIX=/u/example/pathToJarOrZipFile/jarfile.jar:\
#                 /u/example/pathToRootDirectoryForClasses
#
# Elimine el comentario de la línea de debajo para utilizar la clase de redirección
# de salida especificada.
#
#USEROUTPUTCLASS=com.ibm.cics.samples.SJMergedStream
#
######
#
# ********* Variables de entorno de UNIX System Services ***********
#
# Opciones de volcado de Java. Consulte "IBM Developer Kit and Runtime Environment,
# Java Technology Edition, Diagnostics Guide" para obtener información sobre
# todas las opciones de tiempo de ejecución de Java.
JAVA_DUMP_OPTS="ONANYSIGNAL(JAVADUMP,CEEDUMP,SYSDUMP),ONINTERRUPT(NONE)"
```

```
#
# Especifique dónde se deben grabar los volcados de la JVM.
#JAVA_DUMP_TDUMP_PATTERN=DUMP.JVM.TDUMP.&APPLID;.&JVM_NUM;.LATEST
#
# Especifique el huso horario local
#TZ=CET-1CEST,M3.5.0,M10.5.0
#
######
#
# ********* Opciones de JVM **************
#
-Xms16M
-Xmx16M
-Xmso128K
-Xiss64K
-Xss256K
```

DFHJVMCD, perfil de JVM reservado para programas de sistema proporcionados por CICS

El perfil de JVM DFHJVMCD es un perfil de JVM proporcionado por CICS que está reservado para que lo utilicen programas de sistema proporcionados por CICS, en concreto el programa procesador de solicitudes predeterminado, DFJIIRP, que es empleado por la transacción del procesador de solicitudes CIRP proporcionado por CICS. CICS también utiliza DFHJVMCD para inicializar y detener la memoria caché de clase compartida. Asegúrese de que DFHJVMCD esté configurado correctamente para la región CICS, pero solo debe personalizarse si es necesario.

En "Personalización de DFHJVMCD" en la página 96 aparecen instrucciones para personalizar las opciones de este perfil de JVM.

Opciones de JVM en el perfil de JVM DFHJVMCD

```
############################################################
# Perfil de JVM: DFHJVMCD
############################################################
#
# Este es el perfil de JVM de CICS para que lo utilicen los programas de CICS.
# Debe contener un valor válido para JAVA_HOME.
# Siempre debe estar disponible en el directorio especificado por
# el parámetro JVMPROFILEDIR SIT.
#
######
#
# Sustitución de símbolo:
#
# Si utiliza cualquiera de los siguientes símbolos variables en cualquiera de
# las variables de debajo, se sustituirá por los valores
# adecuados. Los símbolos variables pueden especificarse en mayúscula o
# minúscula.
#
#  Símbolo       Valor de sustitución
#  ------       -----------------
#
# &APPLID;      El identificador de aplicación de la región CICS.
# &JVM_NUM;     El ID de proceso (pid) de UNIX System Services
#               de la JVM. Está garantizado que es exclusivo.
# &DATE;        La fecha actual con el formato Daammdd.
# &TIME;        La hora actual con el formato Thhmmss
#
# Con esta sustitución, por ejemplo:
#     STDERR=dfhjvmerr.&APPLID;.&JVM_NUM;.&DATE;.&TIME;
# se convierte en
#     STDERR=dfhjvmerr.ABCDEF.0084214386.D081220.T135323
```

```
#
######
#
# ********* Parámetros específicos de CICS ***********
#
JAVA_HOME=/usr/lpp/java/J6.0.1_64
WORK_DIR=.
REUSE=YES
CLASSCACHE=NO
#
STDIN=dfhjvmin
STDOUT=dfhjvmout
STDERR=dfhjvmerr
######
#
# ********* Variables de entorno de UNIX System Services ***********
#
# Opciones de volcado de Java. Consulte "IBM Developer Kit and Runtime Environment,
# Java Technology Edition, Diagnostics Guide" para obtener información sobre
# todas las opciones de tiempo de ejecución de Java.
JAVA_DUMP_OPTS="ONANYSIGNAL(JAVADUMP,CEEDUMP,SYSDUMP),ONINTERRUPT(NONE)"
#
# Especifique dónde se deben grabar los volcados de la JVM.
#JAVA_DUMP_TDUMP_PATTERN=DUMP.JVM.TDUMP.&APPLID;.%JVM_NUM;.LATEST
#
# Especifique el huso horario local
#TZ=CET-1CEST,M3.5.0,M10.5.0
#
######
#
# ********* Opciones de JVM **************
#
-Xms16M
-Xmx16M
-Xmso128K
-Xiss64K
-Xss256K
```

Capítulo 6. Gestión de aplicaciones Java

Tras haber habilitado las aplicaciones Java, puede supervisar la región CICS para saber cuál es el rendimiento de dichas aplicaciones. También puede ajustar la JVM y el enclave de Language Environment para optimizar el rendimiento de la aplicación.

Acerca de esta tarea

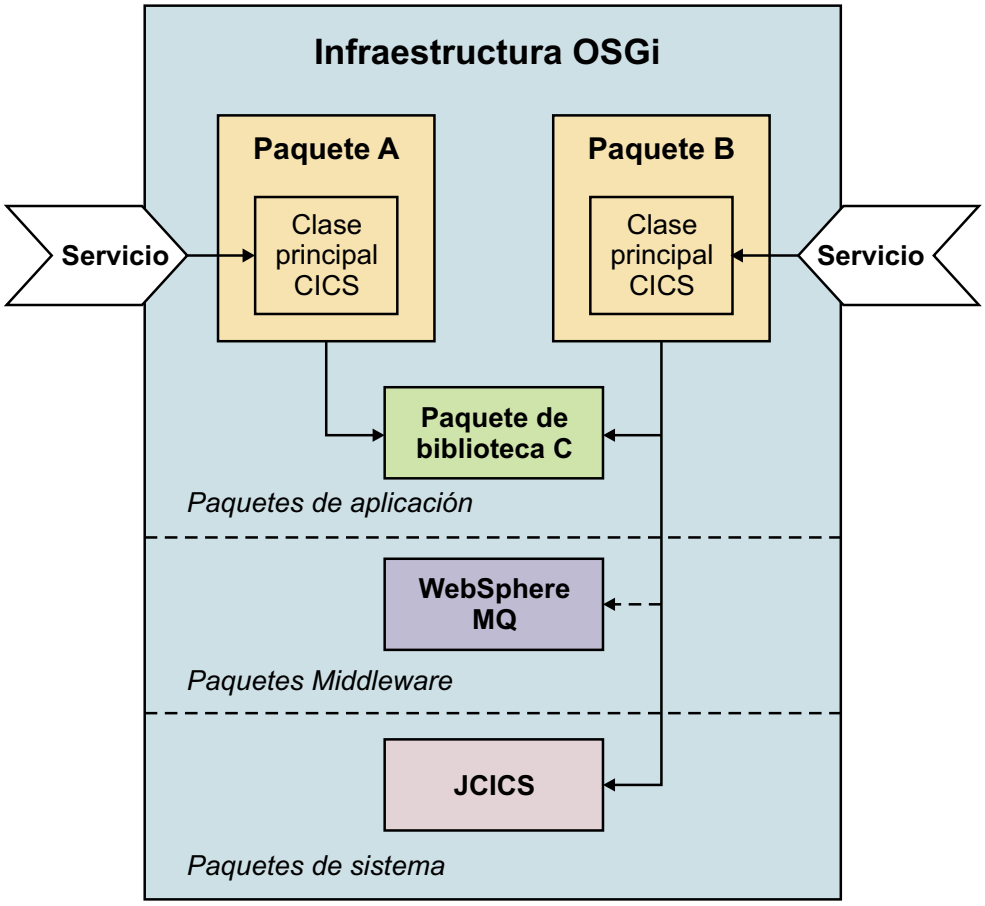
Puede utilizar estadísticas y supervisión para recopilar información sobre cómo es el rendimiento de las aplicaciones Java en la región CICS. En concreto, puede comprobar cómo es el rendimiento de las JVM. Tras recopilar la información, puede realizar cambios en una JVM o en un enclave de Language Environment para mejorar el rendimiento. También puede inhabilitar o mover aplicaciones entre regiones CICS para equilibrar las cargas de trabajo de Java de forma más efectiva.

Actualización de paquetes OSGi en un servidor de JVM

El proceso para actualizar paquetes OSGi en la infraestructura OSGi depende del tipo de paquete y de sus dependencias. Puede actualizar paquetes OSGi para aplicaciones sin reiniciar el servidor de JVM. Sin embargo, actualizar un paquete de middleware requiere un reinicio del servidor de JVM.

Acerca de esta tarea

En un servidor de JVM típico, la infraestructura OSGi contiene una mezcla de paquetes OSGi como se muestra en el siguiente diagrama.



El paquete A y el paquete B son aplicaciones Java distintas que están empaquetadas como paquetes OSGi en paquetes CICS distintos. Ambas aplicaciones tienen una dependencia en una biblioteca común, que está empaquetada en el paquete C. El paquete C se gestiona y se actualiza por separado. Además, el paquete B tiene una dependencia en un paquete de middleware de WebSphere MQ y el el paquete del sistema JCICS.

El paquete A y el paquete B pueden actualizarse de forma independiente sin afectar a ningún otro paquete de la infraestructura. Sin embargo, actualizar el paquete C puede afectar a ambos paquetes que dependen de él. Todos los paquetes exportados en el paquete C permanecen en la memoria del marco OSGi por lo que, para recoger cambios en dicho paquete C, los paquetes A y B también tienen que actualizarse en el marco.

Los paquetes de middleware contienen servicios de infraestructura y se gestionan con el ciclo de vida del servidor de JVM. Por ejemplo, puede contar con código nativo que desee cargar una vez en la infraestructura o tal vez quiera añadir un controlador para cargar a otro producto, como WebSphere MQ.

CICS proporciona paquetes del sistema para gestionar la interacción con la infraestructura OSGi. Estos paquetes cuentan con servicio de IBM como parte del producto. Un ejemplo de un paquete del sistema es el archivo `com.ibm.cics.server.jar`, que proporciona la mayor parte de la API de JCICS para acceder a servicios de CICS.

Actualización de paquetes OSGi

Si un desarrollador de Java proporciona una versión actualizada de un paquete CICS, puede sustituir completamente dicho paquete CICS o puede introducir progresivamente un nueva versión y luego eliminar la antigua.

Antes de empezar

Un paquete CICS actualizado que contenga la nueva versión de un paquete OSGi tiene que estar presente en zFS.

Acerca de esta tarea

Para introducir progresivamente una nueva versión y tener ambos paquetes en ejecución en la infraestructura a la vez, es necesario especificar un alias para el servicio OSGi. Si no se especifica ningún alias, el servicio se lista como inactivo en la infraestructura porque se considera un duplicado del servicio que ya se está ejecutando.

Procedimiento

- Para sustituir el paquete OSGi existente:
  1. Inhabilite y descarte el recurso BUNDLE para el paquete CICS que desea actualizar. Los paquetes y servicios OSGi que forman parte de ese paquete de CICS se eliminan de la infraestructura OSGi.
  2. Opcional: Edite la definición de recurso BUNDLE si el paquete CICS se despliega en un directorio distinto.
  3. Instale la definición de recurso BUNDLE para que detecte el paquete OSGi cambiado. Los paquetes y servicios OSGi del paquete CICS se instalan en la infraestructura OSGi.
  4. Compruebe el estado de los paquetes y servicios OSGi en las vistas **Operations** > **Java** (Operaciones > Java) de CICS Explorer.
- Para crear una nueva versión del paquete OSGi en la infraestructura a la vez que el paquete desplegado existente:
  1. Cree un recurso BUNDLE para que detecte el paquete CICS cambiado. Los paquetes y servicios OSGi del paquete CICS se instalan en la infraestructura OSGi. El servicio OSGi está en estado inactivo, a no ser que se especifique un alias en el manifiesto de paquete.
  2. Compruebe el estado de los paquetes y servicios OSGi en las vistas **Operations** > **Java** (Operaciones > Java) de CICS Explorer. En la vista de paquetes OSGi se listan don versiones del paquete OSGi. El servicio OSGi para el paquete está en estado inactivo, a no ser que se especifique un alias. Si se especifica un alias, ambos servicios OSGi están activos.
  3. Inhabilite el recurso BUNDLE que apunte a la versión antigua del paquete OSGi. CICS elimina el servicio OSGi asociado con el paquete y define el paquete OSGi en el estado resuelto. Como resultado, el servicio OSGi para el paquete OSGi cambiado se mueve de estado inactivo a activo.
  4. Si hay un alias para el servicio OSGi, puede especificar dicho alias en el recurso PROGRAM para llamar a la aplicación actualizada desde fuera del servidor de JVM.

Resultados

La versión simbólica del paquete OSGi se ha incrementado, lo que indica que el código Java se ha actualizado. El paquete OSGi actualizado está disponible en la

infraestructura OSGi y se puede llamar desde el servidor de JVM.

Actualización de paquetes que contienen bibliotecas comunes

Los paquetes OSGi que contienen bibliotecas comunes para que las utilicen otros paquetes OSGi deben actualizarse en un orden específico.

Antes de empezar

Un paquete de CICS actualizado que contenga la nueva versión del paquete OSGi tiene que estar presente en zFS. La práctica recomendada es gestionar las bibliotecas comunes en un paquete de CICS independiente, para poder gestionar el ciclo de vida de las mismas por separado de las aplicaciones que dependen de ellas.

Acerca de esta tarea

Por lo general, un paquete OSGi especifica un rango de versiones soportadas en una dependencia en otro paquete OSGi. Utilizar un rango brinda más flexibilidad para realizar cambios compatibles en la infraestructura. Al actualizar paquetes que contienen bibliotecas comunes, el número de versión del paquete OSGi aumenta. Sin embargo, las aplicaciones en ejecución ya utilizan una versión del paquete que cumple con las dependencias. Para detectar la última versión de la biblioteca, es necesario renovar los paquetes OSGi para las aplicaciones. Por lo tanto, es posible actualizar aplicaciones específicas para que utilicen versiones distintas de la biblioteca y dejar otras aplicaciones ejecutándose en una versión anterior.

Cuando se actualiza un paquete OSGi que contiene bibliotecas comunes, es posible sustituir completamente el paquete CICS. Sin embargo, si no se han cargado clases en la biblioteca, los paquetes dependientes pueden recibir errores. Puede introducir gradualmente una nueva versión de la biblioteca y ejecutarla en la infraestructura junto con la versión original. Siempre que los paquetes OSGi tengan números de versión distintos, la infraestructura OSGi puede ejecutar ambos paquetes a la vez.

Procedimiento

1. Cree un recurso BUNDLE que apunte a la versión nueva del paquete OSGi. CICS crea la nueva versión del paquete OSGi en la infraestructura OSGi. Los paquetes OSGi existentes continúan utilizando la versión anterior de la biblioteca.
2. Consulte la vista de paquetes OSGi en el CICS Explorer. La lista muestra dos entradas para el mismo nombre simbólico de paquete OSGi con versiones distintas ejecutándose en la infraestructura.
3. Para detectar la nueva versión de la biblioteca en una aplicación Java dependiente:
   a. Inhabilite y descarte el recurso BUNDLE para la aplicación Java. Como alternativa, puede hacer que el desarrollador de Java actualice la información sobre la versión del paquete OSGi y despliegue una nueva versión del paquete de CICS para mantener la disponibilidad de la aplicación.
   b. Instale el recurso BUNDLE. Cuando se carga el paquete OSGi en la infraestructura, este detecta la última versión de las bibliotecas comunes.
4. Compruebe el estado del recurso BUNDLE en la vista de paquetes del CICS Explorer.

Resultados

Ha actualizado un paquete OSGi que contiene bibliotecas comunes y ha actualizado una aplicación Java para que utilice la última versión de las bibliotecas.

Actualización de paquetes de middleware de OSGi

Si desea actualizar los paquetes de middleware que se ejecutan en una infraestructura OSGi, debe detener y reiniciar el servidor de JVM.

Acerca de esta tarea

Los paquetes de middleware de OSGi se instalan en la infraestructura OSGi durante la inicialización del servidor de JVM. Si desea actualizar un paquete de middleware (por ejemplo, para aplicar un parche o utilizar una nueva versión), debe detener y reiniciar el servidor de JVM para elegir el paquete cambiado.

Procedimiento

1. Asegúrese de que la nueva versión del paquete de middleware es un directorio en el sistema de archivos zSeries (zFS) al que CICS tiene acceso de lectura y ejecución.CICS también necesita acceso de lectura a los archivos.
2. Si el directorio del zFS o el nombre de archivo es distinto a los valores especificados en el perfil de JVM, edite la opción OSGI_BUNDLES en el perfil de JVM correspondiente al servidor de JVM. Los perfiles de JVM se encuentran en el directorio del zFS especificado por el parámetro de inicialización del sistema **JVMPROFILEDIR**.
3. Inhabilite el recurso JVMSERVER para apagar el servidor de JVM. Inhabilitar JVMSERVER también inhabilita cualquier recurso BUNDLE que contenga paquetes OSGi que estén instalados en ese servidor de JVM.
4. Habilite el recurso JVMSERVER para iniciar el servidor de JVM con el perfil de JVM actualizado. El servidor de JVM se inicia e instala la nueva versión del paquete de middleware en la infraestructura OSGi.CICS también habilita los recursos BUNDLE que se inhabilitaran para instalar los paquetes y los servicios de OSGi en la infraestructura actualizada.

Resultados

La infraestructura OSGi contiene los paquetes de middleware actualizados y los paquetes y servicios de OSGi para aplicaciones Java que se instalaran antes de apagar el servidor de JVM.

Eliminación de paquetes OSGi de un servidor de JVM

Si desea eliminar paquetes OSGi del servidor de JVM, utilice el CICS Explorer para inhabilitar y descartar el recurso BUNDLE.

Acerca de esta tarea

El recurso BUNDLE proporciona gestión de ciclo de vida para la colección de paquetes OSGi y servicios OSGi que se definen en el paquete de CICS. Eliminar paquetes OSGi de la infraestructura OSGi no afecta automáticamente al estado de otros paquetes y servicios OSGi instalados. Si se elimina un paquete que sea un requisito previo para otro paquete, el estado del paquete dependiente no cambia hasta que se renueva explícitamente dicho paquete.

Procedimiento

1. Haga clic en **Operations** > **Java** > **OSGi Bundles** (Operaciones > Java > Paquetes OSGi) para averiguar qué recurso BUNDLE contiene el paquete OSGi.
2. Pulse **Operations** > **Bundles** (Operaciones > Paquetes) para inhabilitar el recurso BUNDLE. CICS inhabilita todos los recursos que están definidos en el paquete de CICS. En el caso de paquetes y servicios OSGi, CICS envía una solicitud a la infraestructura OSGi en el servidor de JVM para anular el registro de cualquier servicio OSGi y mueve los paquetes OSGi a un estado resuelto. Todas las transacciones en curso se completan, pero los nuevos enlaces con el servicio OSGi desde aplicaciones CICS devuelven un error.
3. Descarte el recurso BUNDLE. CICS envía una solicitud a la infraestructura OSGi para eliminar los paquetes OSGi del servidor de JVM.

Resultados

Ha eliminado los paquetes y servicios OSGi de la infraestructura OSGi.

Qué hacer a continuación

Si tienen recursos PROGRAM que apunten a servicios OSGi que ya no están en la infraestructura OSGi, tal vez desee inhabilitar y descartar dichos recursos PROGRAM.

Movimiento de aplicaciones a un servidor de JVM

Si está ejecutando aplicaciones Java en JVM en agrupación, puede moverlas para ejecutarlas en un servidor de JVM. Como un servidor de JVM puede manejar varias solicitudes de aplicaciones Java en la misma JVM, puede reducir el número de JVM que son necesarias para ejecutar la misma carga de trabajo.

Antes de empezar

Asegúrese de que la aplicación es de enhebramiento seguro y de que se empaqueta como uno o más paquetes OSGi. Los paquetes OSGi deben desplegarse en un paquete de CICS en zFS y especificar el recurso de destino JVMSERVER correcto.

El desarrollador de Java puede utilizar el CICS Explorer SDK para volver a empaquetar una aplicación Java mediante OSGi, como se describe en “Migración de aplicaciones utilizando el CICS Explorer SDK” en la página 48.

Acerca de esta tarea

Puede utilizar un servidor de JVM ya existente o crear un servidor de JVM para su aplicación. No mueva una aplicación a un servidor de JVM en el que el límite de hebras y el uso ya sean altos, porque puede crear contenciones de bloqueo en el servidor de JVM.

Procedimiento

1. Cree o actualice un servidor de JVM:
   - Si decide crear un servidor de JVM, consulte “Configuración de un servidor de JVM” en la página 87. Muchos de los valores de un perfil de JVM para una JVM en agrupación no se aplican a los servidores de JVM. La única opción que tal vez desee copiar desde el perfil de JVM agrupada en el perfil DFHOSGI es la opción `LIBPATH_SUFFIX`.

- Si utiliza un servidor de JVM ya existente, tal vez deba aumentar el atributo THREADLIMIT del recurso JVMSERVER para manejar la aplicación adicional o actualizar las opciones en el perfil de servidor de JVM. Cualquier cambio en el recurso JVMSERVER o en el perfil de JVM requieren el reinicio del servidor de JVM para detectar dichos cambios.

2. Cree un recurso BUNDLE que apunte al paquete desplegado en zFS. Cuando se instala el recurso BUNDLE, CICS carga los paquetes OSGi en la infraestructura OSGi del servidor de JVM. La infraestructura OSGi resuelve los paquetes OSGi y registra los servicios OSGi. Utilice el CICS Explorer para comprobar que el recurso BUNDLE está habilitado. También puede utilizar las vistas OSGi Bundles (Paquetes OSGi) y OSGi Services (Servicios OSGi) para comprobar el estado de los paquetes y servicios OSGi.
3. Actualice el recurso PROGRAM para la aplicación:
   a. Asegúrese de que el atributo EXECKEY está definido para CICS. Todo el trabajo del servidor de JVM se ejecuta en clave de CICS.
   b. Elimine el nombre del perfil de JVM y escriba el nombre del recurso JVMSERVER.
   c. Asegúrese de que el atributo JVMCLASS coincide con el servicio OSGi de la aplicación Java.
   d. Reinstale el recurso PROGRAM para la aplicación.

   El recurso PROGRAM utiliza el servicio OSGi para hacer que un paquete OSGi esté disponible para otras aplicaciones CICS fuera del servidor de JVM.

Resultados

Cuando se llama a la aplicación Java, esta se ejecuta en el servidor de JVM.

Qué hacer a continuación

Puede utilizar la vista de servidores de JVM del CICS Explorer y estadísticas de CICS para supervisar el servidor de JVM. Si el rendimiento no es óptimo, ajuste el límite de hebras.

Gestión del límite de hebras de los servidores de JVM

Los servidores de JVM tienen limitado el número de hebras que pueden utilizar para ejecutar aplicaciones Java. La región CICS también tiene un límite al número de hebras, porque cada hebra utiliza un TCB T8. Puede ajustar el límite de hebras utilizando estadísticas de CICS para equilibrar el número de servidores de JVM de la región según el rendimiento de las aplicaciones que se ejecutan en cada servidor de JVM.

Acerca de esta tarea

Cada servidor de JVM puede tener un máximo de 256 hebras para ejecutar aplicaciones Java. En una región CICS, puede tener un máximo de 1024 hebras. Si tiene muchos servidores de JVM ejecutándose en la región CICS, no se puede definir el valor máximo para cada servidor de JVM. Si define el valor máximo en cuatro servidores de JVM, no puede habilitar ningún otro recurso JVMSERVER en la región CICS. Puede ajustar el límite de hebras de cada servidor de JVM para equilibrar el número de servidores de JVM de la región CICS según el rendimiento de las aplicaciones Java.

El límite de hebras se define en el recurso JVMSERVER, así que defina un valor inicial y utilice estadísticas de CICS para ajustar el número de hebras cuando esté probando cargas de trabajo de Java.

Procedimiento

1. Habilite los recursos JVMSERVER y ejecute la carga de trabajo de la aplicación Java.
2. Recopile estadísticas de recurso JVMSERVER utilizando un intervalo de estadísticas adecuado. Puede utilizar la vista **Operations** > **Java** > **JVM Servers** (Operaciones > Java > Servidores de JVM) de CICS Explorer, o puede utilizar el programa de estadísticas DFH0STAT.
3. Compruebe cuántas veces y cuánto tiempo una tarea tuvo que esperar una hebra. Los campos "JVMSERVER thread limit waits" (Esperas de límite de hebra) y "JVMSERVER thread limit wait time" (Veces de espera de límite de hebras) contienen esta información.
   - Si los valores de estos campos son altos y hay muchas tareas suspendidas con la espera de JVMTHRD, el servidor de JVM no tiene suficientes hebras disponibles. Aumentar el número de hebras puede aumentar el uso de procesador, así que compruebe que tiene suficientes recursos de MVS disponibles.
   - Si los valores de estos campos son bajos y el número máximo de tareas es inferior al número máximo de hebras disponibles, puede liberar hebras para otros servidores de JVM reduciendo el límite de hebras.
4. Para comprobar la disponibilidad de recurso de MVS, utilice la agrupación de TCB del asignador y las estadísticas de modalidad de TCB para evaluar el uso de TCB T8 en la región CICS. Cada hebra de un servidor de JVM utiliza un TCB T8 y el límite es de 1024 en una región. Los TCB no se pueden compartir entre servidores de JVM, aunque todos los TCB estén en una misma agrupación de TCB THRD. Si el número de TCB en espera y el uso de procesador es bajo, esto indica que hay suficiente recurso de MVS disponible.
5. Para ajustar el número de hebras que se pueden ejecutar en el servidor de JVM, cambie el atributo THREADLIMIT y vuelva a habilitar el recurso JVMSERVER.
6. Ejecute la carga de trabajo de la aplicación Java otra vez y utilice las estadísticas para comprobar que el número de tareas en espera se ha reducido.

Qué hacer a continuación

Para ajustar el rendimiento de los servidores de JVM, consulte "Mejora del rendimiento del servidor de JVM" en la página 167.

Recuperación del paquete OSGi en un reinicio de CICS

Cuando reinicie una región CICS que contenga paquetes OSGi, CICS recuperará los recursos BUNDLE e instalará los paquetes OSGi en la infraestructura del servidor de JVM.

Los paquetes OSGi que están empaquetados en paquetes de CICS no se almacenan en CSD. El propio recurso BUNDLE se almacena en el catálogo, para que durante el reinicio de la región CICS, los paquetes OSGi se vuelvan a crear dinámicamente cuando se restaure el recurso BUNDLE.

En un reinicio de emergencia, en frío o en caliente de CICS, el servidor de JVM se inicia de forma asíncrona para la recuperación del recurso BUNDLE. El servidor de JVM debe estar plenamente disponible para restaurar de forma satisfactoria un

paquete OSGi cuando se reinicia CICS. Por lo tanto, aunque los recursos BUNDLE se recuperan durante la última fase del inicio de CICS, los paquetes OSGi se instalan únicamente cuando el servidor de JVM ha finalizado su lanzamiento.

Los recursos BUNDLE y los paquetes de OSGi que contienen se instalan en el orden correcto para asegurarse de que las dependencias entre los paquetes de CICS y de OSGi se resuelvan en la infraestructura. Si CICS no se instala como un paquete de OSGi, el recurso BUNDLE se instala con el estado de inhabilitado. Puede utilizar IBM CICS Explorer para ver el estado de los recursos BUNDLE, los paquetes OSGi y los servicios OSGi.

Actualización de aplicaciones Java en JVM en agrupación

Si cambia aplicaciones Java que se ejecutan en JVM en agrupación, debe detener y reiniciar las JVM que ejecuten dichas aplicaciones para cargar los recursos cambiados. También debe detener y reiniciar las JVM en agrupación si realiza cualquier cambio en los recursos o en los archivos en la vía de acceso de clases.

Antes de empezar

La aplicación Java actualizada debe compilarse, empaquetarse y desplegarse en el sistema de archivos z/OS UNIX.

Acerca de esta tarea

Puede añadir clases de aplicación Java a las vías de acceso de clases para una JVM, o puede cambiar los nombres de los archivos. Cuando se están ejecutando, las JVM no reconocen cambios en los perfiles de JVM, por lo que es necesario detener y reiniciar las JVM para detectar los cambios en la aplicación.

Procedimiento

1. Opcional: Edite el perfil de JVM correspondiente a la aplicación para añadir las clases nuevas o cambiadas a la vía de acceso de clases. Si ha cambiado el contenido de una clase o de un archivo JAR pero conserva el mismo nombre, no es necesario realizar este paso.
2. Reinicie la JVM para detectar los cambios en la aplicación. Desactive la agrupación de JVM para cada perfil de JVM que indique el archivo cambiado. Las JVM que no ejecuten esta aplicación pueden seguir ejecutándose. Si hay solicitudes esperando JVM con los perfiles que ha desactivado, CICS lanza nuevas JVM. La memoria caché de clase compartida se actualiza automáticamente cuando la JVM carga las clases cambiadas, por lo que no tiene que reiniciarla.

Resultados

CICS crea una JVM en agrupación utilizando la versión actualizada del perfil de JVM y carga las clases nuevas o cambiadas.

Escritura de clases Java para redirigir las salidas stdout y stderr de JVM

Utilice la opción `USEROUTPUTCLASS` de un perfil de JVM para indicar una clase Java que intercepte las salidas stdout y stderr desde la JVM. Puede actualizar esta clase para especificar las indicaciones de fecha y hora y las cabeceras de registro que elija y para redirigir la salida.

CICS proporciona clases Java de ejemplo, com.ibm.cics.samples.SJMergedStream y com.ibm.cics.samples.SJTaskStream, que puede utilizar para este fin. Se ofrece código fuente para ambas clases en el directorio `/usr/lpp/cicsts/cicsts42/samples/com.ibm.cics.samples`. El directorio `/usr/lpp/cicsts/cicsts42` es el directorio de instalación para archivos de CICS en z/OS UNIX. Este directorio se especifica mediante el parámetro **USSDIR** en el trabajo de instalación DFHISTAR. Las clases de ejemplo también se incluyen como un archivo de clase, `com.ibm.cics.samples.jar`, que se encuentra en el directorio `/usr/lpp/cicsts/cicsts42/lib`. Puede modificar dichas clases o escribir las suyas propias basándose en los ejemplos.

En "Control de la ubicación para las salidas de JVM stdout, stderr y de volcado" en la página 197 se muestra información acerca de:

- Los tipos de salida desde las JVM que intercepta y que no intercepta la clase indicada por la opción `USEROUTPUTCLASS`. La clase que utilice debe ser capaz de gestionar todos los tipos de salida que pueda interceptar.
- El comportamiento de las clases de ejemplo proporcionadas. La clase com.ibm.cics.samples.SJMergedStream crea dos archivos de registro fusionados para la salida de la JVM y para los mensajes de error, con una cabecera en cada registro que contiene el ID de aplicación, la fecha, la hora, el ID de transacción, el número de tarea y el nombre de programa. Los archivos de registro se crean utilizando colas de datos transitorias, si están disponibles, o archivos z/OS UNIX, si no hay colas de datos transitorias disponibles o si la aplicación Java no puede utilizarlas. La clase com.ibm.cics.samples.SJTaskStream dirige la salida desde una única tarea a archivos z/OS UNIX, añadiendo indicaciones de fecha y hora y cabeceras, para proporcionar secuencias de salida específicas para una tarea individual.

Para que una JVM agrupada utilice una clase de redirección de salida que haya modificado o escrito, la clase debe estar presente en un directorio de una vía de acceso de clases adecuada en el perfil de JVM o en el archivo de propiedades. El directorio que contiene el archivo JAR para la clase de redirección de salida de ejemplo se incluye automáticamente en una vía de acceso de clases adecuada y no tiene que especificarla explícitamente en el perfil de JVM. Si proporciona su propia clase, tendrá que añadir el directorio a la vía de acceso de clase estándar.

Con un servidor de JVM, para utilizar la clase de redirección de salida debe crear un paquete OSGi que contenga la clase de redirección de salida. Debe asegurarse de que el activador de paquetes registre una instancia de la clase como un servicio en la infraestructura y configure la propiedad `com.ibm.cics.server.outputredirectionplugin.name=`*`nombre_clase`*. Puede utilizar la constante `com.ibm.cics.server.Constants.CICS_USER_OUTPUT_CLASSNAME_PROPERTY` para obtener el nombre de propiedad. El siguiente fragmento de código muestra cómo puede registrar el servicio en el activador de paquetes:

```
Properties serviceProperties = new Properties();
    serviceProperties.put(Constants.CICS_USER_OUTPUT_CLASSNAME_PROPERTY,
    MyOwnStreamPlugin.class.getName());
    context.registerService(OutputRedirectionPlugin.class.getName(),
    new MyOwnStreamPlugin(), serviceProperties);
```

También puede añadir el paquete OSGi a la opción `OSGI_BUNDLES` en el perfil de JVM o asegurarse de que el paquete se instale en la infraestructura cuando se ejecute la primera tarea. Independientemente del método que utilicde, tendrá que especificar la clase en la opción `USEROUTPUTCLASS`.

Si decide escribir sus propias clases, tiene que tener conocimiento sobre:

- La interfaz OutputRedirectionPlugin
- Destinos posibles de la salida
- Manejo de los errores de redirección de salida y de los errores internos

Interfaz de redirección de salida

CICS proporciona una interfaz llamada com.ibm.cics.server.OutputRedirectionPlugin en `com.ibm.cics.server.jar`, que pueden implementar las clases que interceptan la salida stdout y stderr de la JVM. Los ejemplos que se proporcionan implementan esta interfaz.

Se proporcionan las siguientes clases de ejemplo:

- Una superclase com.ibm.cics.samples.SJStream que implementa esta interfaz.
- Las subclases com.ibm.cics.samples.SJMergedStream y com.ibm.cics.samples.SJTaskStream, que son las clases que se indican en el perfil de JVM

Al igual que sucede con las clases de ejemplo, asegúrese de que su clase implementa la interfaz OutputRedirectionPlugin directamente o de que amplía una clase que implemente la interfaz. Puede heredar de la superclase com.ibm.cics.samples.SJStream, o implementar una estructura de clases con la misma interfaz. Sea cual sea el método que utilice, la clase debe ampliar java.io.OutputStream.

El método initRedirect() recibe un conjunto de parámetros que son empleados por la clase o las clases de redirección de salida. El siguiente código muestra la interfaz:

```
package com.ibm.cics.server;

import java.io.*;

public interface OutputRedirectionPlugin {

  public boolean initRedirect( String inDest,
                               PrintStream inPS,
                               String inApplid,
                               String inProgramName,
                               Integer inTaskNumber,
                               String inTransid
                              );
  }
```

La superclase com.ibm.cics.samples.SJStream contiene los componentes comunes de com.ibm.cics.samples.SJMergedStream y com.ibm.cics.samples.SJTaskStream. Contiene un método initRedirect() que devuelve `false`, lo que inhabilita de forma efectiva la redirección de salida a no ser que este método se sustituya por otro método en una subclase. No implementa un método writeRecord(), y dicho método debe proporcionarlo alguna subclase para controlar el proceso de redirección de salida. Puede utilizar este método en su propia estructura de clases. La inicialización de la redirección de salida también se puede realizar utilizando un constructor en lugar del método initRedirect().

El parámetro **`inPS`** contiene la secuencia de impresión System.out original o la secuencia de impresión System.err de la máquina virtual Java. Puede grabar el registro en cualquiera de estos destinos de registro subyacentes. No debe llamar al

método close() en ninguna de estas secuencias de impresión porque permanecen cerradas permanentemente y no están disponibles para volver a utilizarlas.

Destinos posibles de la salida

Las clases de ejemplo proporcionadas por CICS dirigen la salida desde las JVM a un directorio que es específico para la región CICS. El nombre del directorio se crea utilizando el identificador de aplicación asociado con la región CICS. Cuando escriba sus propias clases, si lo prefiere, puede enviar salida desde varias regiones CICS al mismo directorio o archivo z/OS UNIX.

Por ejemplo, tal vez desee crear un único archivo que contenga la salida asociada con una aplicación concreta que se ejecuta en varias regiones CICS distintas.

Las aplicaciones Java que se ejecutan en hebras distintas a la hebra de proceso inicial (IPT) no pueden realizar solicitudes de CICS. En el caso de estas aplicaciones, la salida de la JVM es interceptada por la clase que haya especificado en USEROUTPUTCLASS, pero no se puede redirigir mediante recursos de CICS (como colas de datos transitorias). Puede dirigir la salida desde estas aplicaciones a archivos z/OS UNIX, como hacen las clases de ejemplo que se proporcionan. En el caso de aplicaciones Java que se ejecutan en la IPT, puede utilizar recursos de CICS, como colas de datos transitorias, para redirigir la salida.

Manejo de los errores de redirección de salida y de los errores internos

Si sus clases utilizan recursos de CICS para redirigir la salida, deberían incluir el manejo de excepciones adecuado para gestionar los errores derivados del uso de dichos recursos.

Por ejemplo, si graba en las colas de datos transitorios CSJO y CSJE y utiliza las definiciones proporcionadas por CICS para dichas colas, TDQ.writeData puede generar las siguientes excepciones:

- IOErrorException
- LengthErrorException
- NoSpaceException
- NotOpenException

Si sus clases dirigen la salida a archivos z/OS UNIX, deberían incluir el manejo de excepciones adecuado para gestionar los errores que se produzcan al grabar en z/OS UNIX. La causa más común de estos errores es una excepción de seguridad.

Los programas Java que se ejecutarán en JVM que especifican sus clases en las opciones de USEROUTPUTCLASS deberían incluir el manejo de excepciones adecuado para gestionar las excepciones que puedan generar las clases. Las clases de ejemplo proporcionadas por CICS manejan las excepciones de forma interna, mediante un bloque Try/Catch para detectar todas las excepciones que se puedan generar y luego grabar uno o más mensajes de error para informar del problema. Cuando se detecta un error al redirigir un mensaje de salida, estos mensajes de error se graban en `System.err`, con lo que están disponibles para redirección. No obstante, si se encuentra un error al redirigir un mensaje de error, los mensajes que informan de este problema se graban en el archivo indicado por la opción STDERR en el perfil de JVM utilizado por la JVM que atiende la solicitud. Como las clases de ejemplo detectan todos los errores de esta manera, los programas de llamada no tienen que manejar ninguna excepción generada por la clase de redirección de salida. Puede utilizar este método para evitar realizar cambios en sus programas

de llamada. Tenga cuidado de no enviar la clase de redirección de salida a un bucle al intentar redirigir el mensaje de error emitido por la clase al destino que ha sufrido la anomalía.

Gestión de JVM en agrupación

CICS realiza muchas tareas para gestionar JVM en agrupación, lo que incluye la creación y la reutilización de JVM. Para optimizar el rendimiento, puede supervisar las JVM en agrupación y la memoria caché de clase compartida, así como ajustar el entorno Java.

Puede iniciar y detener algunas JVM de la agrupación de JVM o inhabilitar la agrupación en la región CICS. También puede ajustar las opciones en los perfiles de JVM, por ejemplo, cambiando el umbral de tiempo de espera para determinar cuánto espera CICS antes de eliminar las JVM inactivas de la agrupación.

CICS proporciona estadísticas e información de supervisión sobre el comportamiento de las JVM en agrupación en la región CICS. Puede utilizar esta información para ayudar a ajustar el entorno Java y optimizar el rendimiento. Para obtener más información sobre cómo alcanzar el rendimiento óptimo, consulte Capítulo 7, “Mejora del rendimiento de Java”, en la página 161.

Cómo asigna CICS JVM en agrupación a las aplicaciones

Cuando una aplicación ejecuta un programa Java que se ejecuta utilizando una JVM en agrupación, primero CICS intenta encontrar una JVM adecuada que esté disponible para reutilizarse en la agrupación de JVM. Si no hay disponible ninguna JVM adecuada, con el perfil de JVM y la clave de ejecución correctas, CICS crea una JVM, si es posible, o utiliza su mecanismo de selección para decidir un curso de acción alternativo.

Una aplicación puede reutilizar una JVM en agrupación disponible si la JVM se creó utilizando el perfil de JVM y la clave de ejecución (USER o CICS) que se especifican en el recurso PROGRAM correspondiente al programa Java. Si hay disponible una JVM adecuada, CICS asigna la JVM a la solicitud.

Si no hay disponible ninguna JVM adecuada, con el perfil de JVM y la clave de ejecución correctas, el límite definido por el parámetro de inicialización del sistema **MAXJVMTCBS** aún no se ha alcanzado y el almacenamiento de MVS no está muy restringido, CICS crea una nueva JVM para el programa Java. La nueva JVM tiene el perfil y la clave de ejecución correctas para el programa.

Si CICS no puede encontrar una JVM adecuada y no se puede crear una nueva JVM porque se ha alcanzado el límite de **MAXJVMTCBS**, o porque el almacenamiento MVS está muy restringido y CICS actúa como si se hubiera alcanzado el límite de **MAXJVMTCBS**, CICS debe decidir el mejor modo de proporcionar una JVM a la aplicación. Esto implica evaluar cuánto necesita la aplicación una JVM frente a la necesidad de distintos tipos de JVM en la región CICS. CICS puede satisfacer la solicitud para obtener una JVM realizada por una aplicación de uno de estos modos:

- Toma para la solicitud una JVM libre que tenga la clave de ejecución correcta pero el perfil incorrecto, destruyendo la JVM y volviéndola a crear en el TCB de la JVM antigua con el perfil correcto. Esto se conoce como una *no coincidencia*.
- Destruye una JVM y su TCB que se encuentran en la clave de ejecución incorrecta y los sustituye por una JVM y un TCB en la clave de ejecución

correcta. Esta situación se conoce como un *robo*, o *robar*, ya que se ha "robado" el TCB de una modalidad de TCB (J8 o J9) para otra modalidad de TCB.

Tanto una no coincidencia como un robo son caros, por lo que antes de adoptar uno de estos cursos de acción, CICS intenta decidir si merece la pena. En términos de la necesidad de distintos tipos de JVM en la región CICS, puede resultar más económico para el rendimiento del sistema en general que CICS haga esperar a la aplicación hasta que haya disponible una JVM adecuada y mantener las JVM libres para solicitudes que puedan beneficiarse más de ellas. CICS dispone de un mecanismo de selección para tomar esta decisión.

La Figura 4 muestra cómo se desarrolla este proceso.

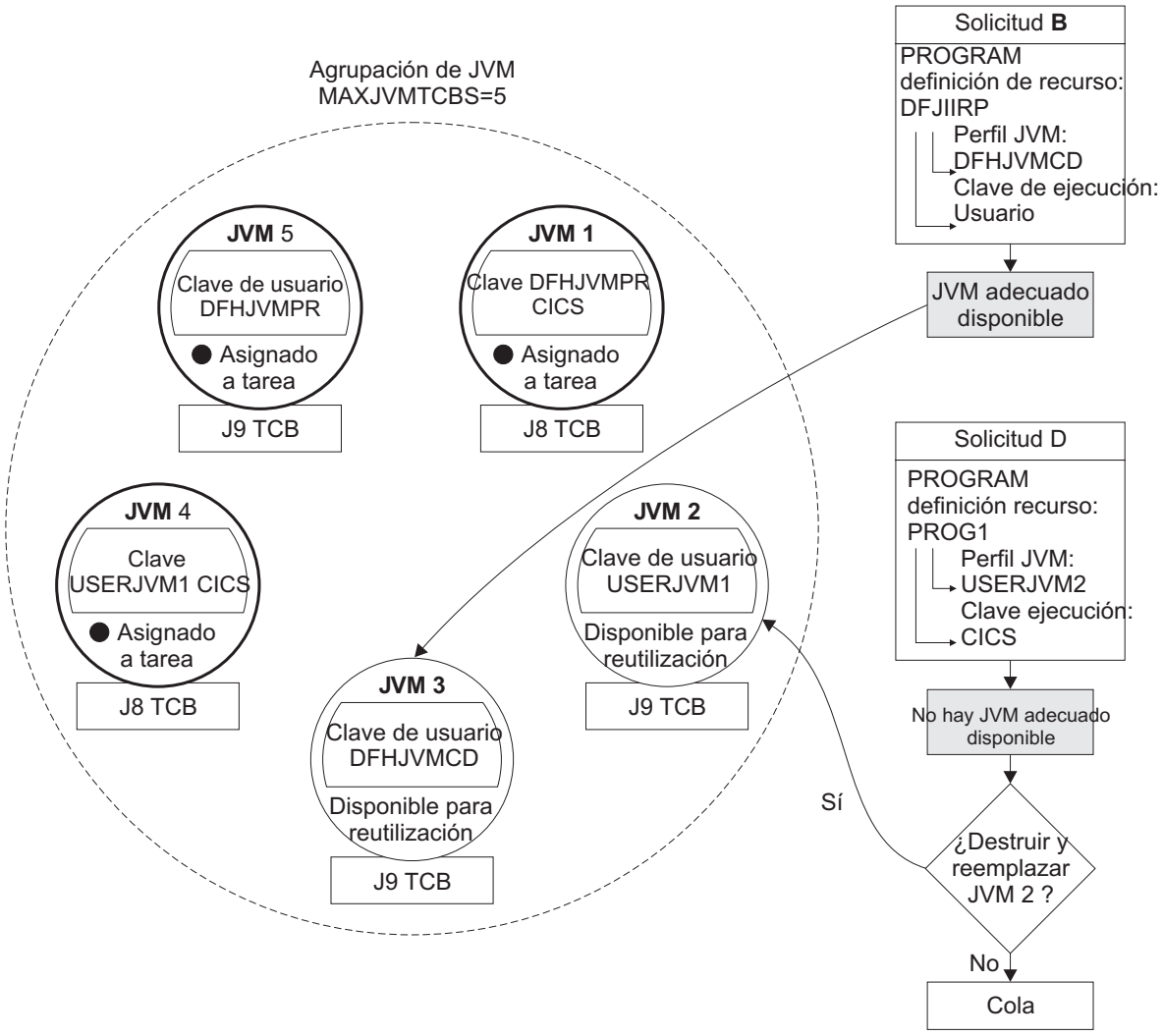


Figura 4. Gestión de solicitudes de JVM: ejemplo

La solicitud B especifica la definición de recurso PROGRAM para el programa procesador de solicitudes predeterminado DFJIIRP, que indica el perfil de JVM DFHJVMCD y la clave de ejecución USER. CICS comprueba la agrupación de JVM y encuentra que la JVM 3 tiene el perfil de JVM y la clave de ejecución correctas para coincidir con la solicitud y está disponible para su reutilización. CICS asigna la JVM 3 a la solicitud B.

La solicitud D especifica la definición de recurso PROGRAM para PROG1, que indica el perfil de JVM USERJVM2 y la clave de ejecución CICS. CICS comprueba la agrupación de JVM. Hay una JVM libre, JVM 2, pero tiene el perfil y la clave de ejecución incorrectas para la solicitud D. Como se ha alcanzado el límite **MAXJVMTCBS**, CICS no puede crear una nueva JVM para la solicitud D. Por tanto CICS debe utilizar el mecanismo de selección para decidir si debe destruir JVM 2 y su TCB y los sustituye por una JVM y un TCB que coincidan con la solicitud D, o si tiene que hacer que la solicitud D espere y mantener la JVM 2 para una solicitud que se pueda beneficiar más de ella. Si se hace esperar a la solicitud D, se pone en cola con cualquier otra solicitud que espere una JVM.

CICS toma su decisión sobre la asignación de una JVM a una aplicación en dos fases:

- Se necesita un conjunto de acciones para gestionar las solicitudes entrantes de JVM.
- Se necesita otro conjunto de acciones distinto cuando se tiene una cola de solicitudes que esperan una JVM.

Cómo gestiona CICS las solicitudes entrantes de JVM

Para gestionar las solicitudes entrantes para obtener una JVM, CICS realiza las acciones resumidas a continuación.

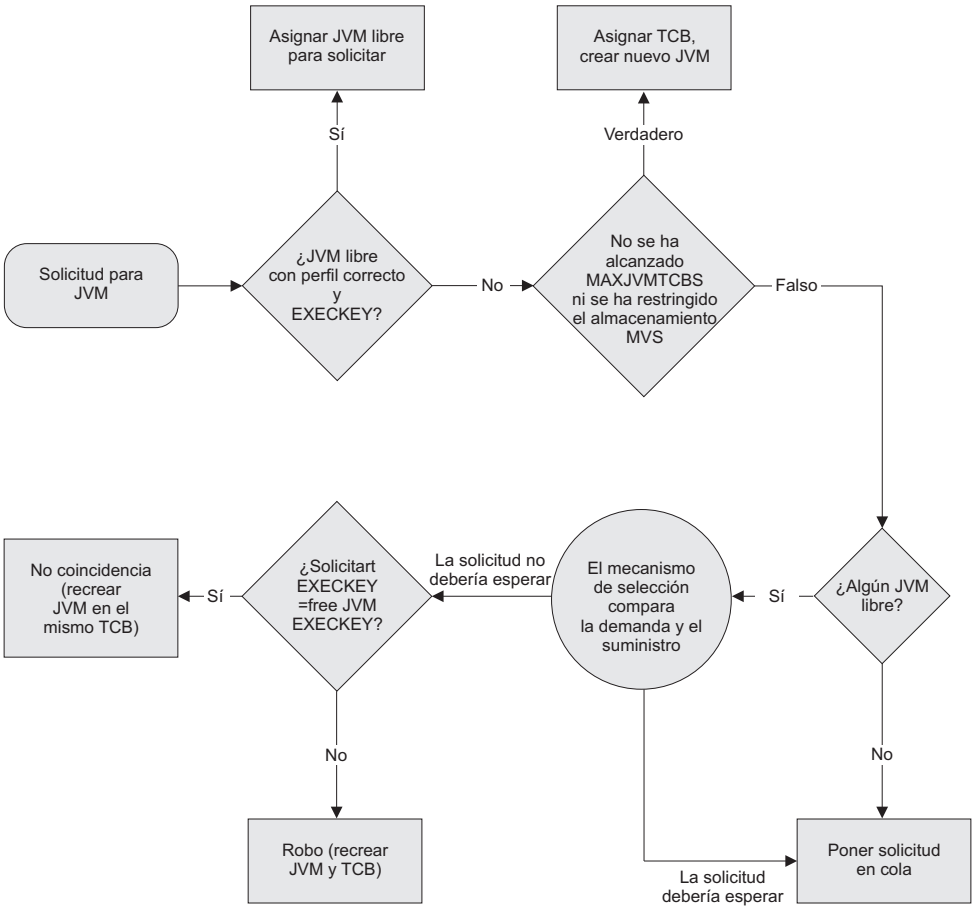


Figura 5. Gestión de solicitudes entrantes de JVM

1. Cuando CICS recibe una solicitud para obtener una JVM y hay una JVM con el perfil y la clave de ejecución correctas, CICS asigna la JVM a la solicitud entrante.
2. Si CICS recibe una solicitud de una JVM cuando:
   - No hay JVM libres.
   - O bien hay JVM libres pero no tienen el perfil y la clave de ejecución correctas para la solicitud.

   Y CICS **puede** crear más JVM (porque el límite de **MAXJVMTCBS** no se ha alcanzado y el almacenamiento MVS no está gravemente restringido), se asigna un TCB y se crea una nueva JVM para la solicitud.
3. Si CICS recibe una solicitud cuando hay JVM libres, pero estas no tienen el perfil y la clave de ejecución correctas y CICS **no** puede crear más JVM (porque el límite de **MAXJVMTCBS** se ha alcanzado o el almacenamiento MVS está gravemente restringido), se utiliza el mecanismo de selección. El mecanismo de selección decide si la solicitud debe esperar a una JVM adecuada o si debe recibir una de las JVM libres.
   a. Si la solicitud recibe una de las JVM libres, se producirá una no coincidencia o un robo, y la JVM y probablemente el TCB tendrán que reinicializarse, por lo que el mecanismo de selección evita hacer esto siempre que sea posible. Si el mecanismo de selección decide que la solicitud sí debe recibir una de las JVM libres, CICS comprueba si la clave de ejecución especificada por la solicitud coincide con la clave de ejecución de la JVM. Si la clave de ejecución no coincide, la JVM y su TCB se destruyen y reinicializan (un robo). Si la clave de ejecución sí coincide y lo único incorrecto es el perfil de JVM, la JVM se vuelve a inicializar en el mismo TCB (una no coincidencia).
   b. Si el mecanismo de selección decide que la solicitud debe esperar en lugar de recibir una de las JVM libres, la solicitud se pone en la cola en espera de que quede libre una JVM adecuada.
4. Si CICS recibe una solicitud cuando no hay JVM libres, y CICS **no** puede crear más JVM (porque se ha alcanzado el límite de **MAXJVMTCBS** o el almacenamiento MVS está gravemente restringido), la solicitud se pone en la cola en espera de que quede libre una JVM.

Cómo gestiona CICS una cola de solicitudes que esperan una JVM

Si CICS tiene una cola de solicitudes que esperan una JVM, realiza las siguientes acciones.

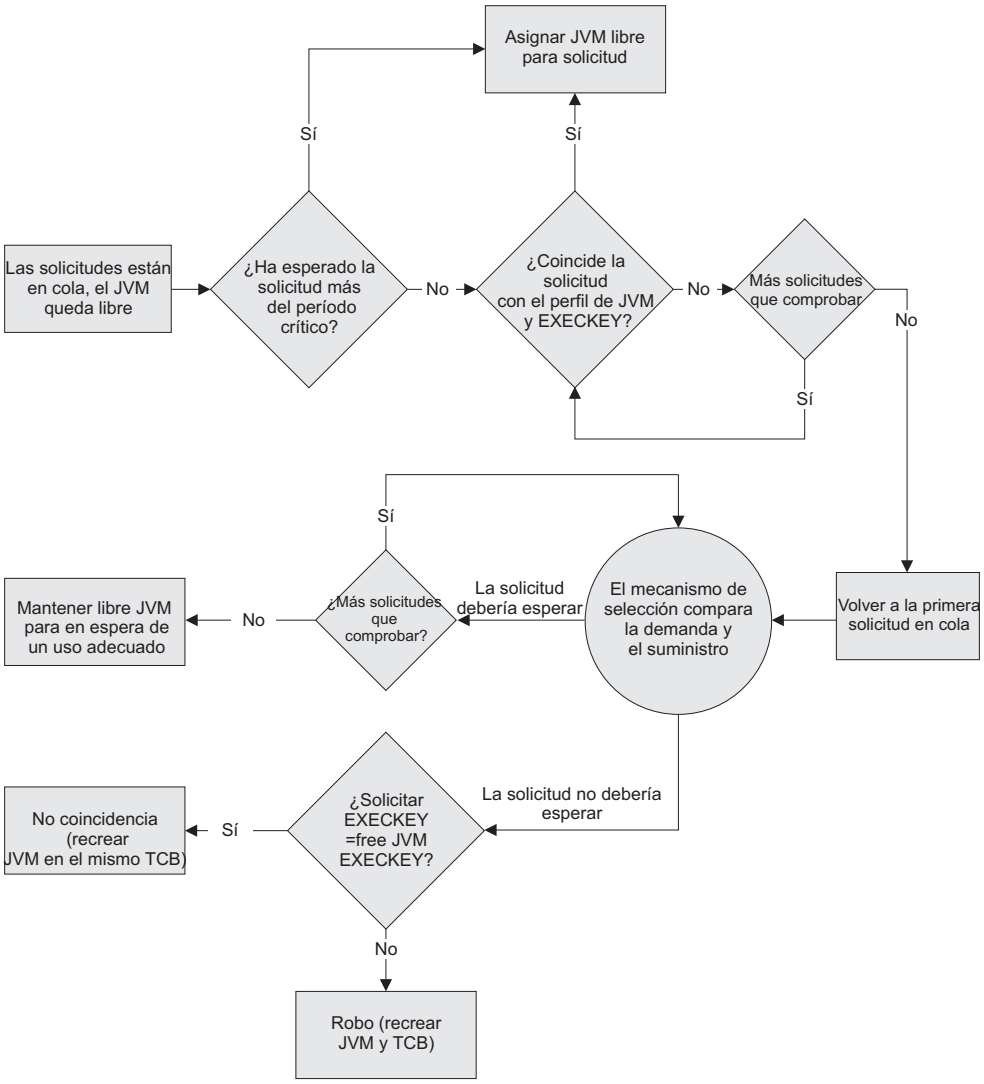


Figura 6. Gestión de una cola de solicitudes que esperan una JVM

1. Si cualquier solicitud que esté esperando a que una JVM quede libre ha esperado más de un periodo crítico (que CICS determina), CICS le concede la siguiente JVM disponible, independientemente del perfil y de la clave de ejecución de dicha JVM. Esto se aplica tanto a solicitudes que se han colocado en la cola porque no había JVM libres como a solicitudes que se han colocado en la cola porque las JVM libres tienen el perfil o la clave de ejecución incorrecta. Se producirá una no coincidencia o un robo, y es posible que la JVM y probablemente el TCB se vuelvan a inicializar (a no ser que la solicitud esté en una cola y que la siguiente JVM libre tenga el perfil y la clave de ejecución correctas), pero la acción merece la pena, ya que la solicitud no debe esperar más.
2. Si las solicitudes están en cola y una JVM queda libre, pero ninguna solicitud ha estado esperando más que el periodo crítico, CICS explora la cola para encontrar la solicitud que más ha esperado y que necesite una JVM con ese perfil y esa clave de ejecución. Concede la JVM libre a la solicitud que más ha

esperado y que especifique el perfil y la clave de ejecución correctas. Por lo tanto, en esta situación, la JVM no se tiene que reinicializar y se evita una no coincidencia o un robo.

3. Si CICS no puede encontrar ninguna solicitud que coincida con el perfil y la clave de ejecución de la JVM libre, vuelve a explorar la cola y utiliza el mecanismo de selección para buscar una solicitud en la que sería una ventaja destruir y reinicializar la JVM libre y reinicializarla como una JVM con el perfil y la clave de ejecución que necesita la solicitud. Se produce una no coincidencia o un robo, pero el mecanismo de selección se asegura de que se produzca para una solicitud que lo merezca.
4. Si CICS no encuentra ninguna solicitud en la cola en la que sería una ventaja destruir y reinicializar la JVM libre, la JVM se mantiene libre para esperar un uso más adecuado. Por ejemplo, CICS podría recibir una solicitud que necesitara una JVM con el perfil y la clave de ejecución de la JVM libre. O la primera solicitud de la cola podría haber esperado más que el periodo crítico y, por lo tanto, se le concedería la JVM libre. O CICS podría recibir una solicitud en la que fuera una ventaja destruir y reinicializar la JVM libre.

El mecanismo de selección

El mecanismo de selección se utiliza cuando CICS tiene que conocer si una solicitud entrante debe esperar a una JVM más adecuada, o si CICS tiene una cola de solicitudes que no coinciden con ninguna JVM libre y tiene que saber si alguna de ellas merece tomar, destruir y reinicializar la JVM.

En estas situaciones, el mecanismo tiene en cuenta la imagen completa de la necesidad de distintos tipos de JVM en la región CICS. Compara la demanda y el suministro de JVM con cada perfil y clave de ejecución mediante el análisis de:

- Los datos históricos relacionados con solicitudes recientes de cada tipo de JVM (la demanda).
- El número de cada tipo de JVM de la agrupación y el tiempo durante el cuál las tareas mantienen estas JVM (el suministro).

El mecanismo de selección utiliza estos datos para averiguar si una determinada solicitud debe esperar a una JVM con el perfil y la clave de ejecución correctas, o si debe concedérsele una JVM libre. La misma respuesta es válida para una solicitud que espere en una cola a que una JVM quede libre o para una solicitud que se realice cuando hay JVM libres pero estas no tengan el perfil o la clave de ejecución correcta. En ambos casos, se hace que la solicitud espere si los datos indican que la demanda del tipo de JVM (es decir, una JVM con ese perfil y esa clave de ejecución) que quiere la solicitud es generalmente *inferior* al suministro y, por lo tanto, no merece la pena destruir y recrear la JVM libre como una JVM de ese tipo. Cuando el mecanismo de selección examina una cola de solicitudes, desciende por dicha cola hasta que alcanza una solicitud en la que los datos indiquen que la demanda de ese tipo de JVM que quiere la solicitud generalmente es *superior* al suministro. Para esta solicitud, el mecanismo de selección decide que, debido a que se necesitan JVM de ese tipo en la región CICS, merece la pena destruir y recrear la JVM libre como una JVM de ese tipo y asigna la JVM libre a la solicitud. Si la JVM libre tenía el perfil incorrecto pero la clave de ejecución correcta, se trata de una no coincidencia y la JVM se reinicializa. Si la JVM libre tenía la clave de ejecución incorrecta pero el perfil correcto, se trata de un robo y tanto el TCB como la JVM se destruyen y se recrean. Así que, aunque se sigue produciendo la sobrecarga de reinicializar la JVM y, si es necesario, recrear el TCB, el mecanismo de selección se ha asegurado de que la nueva JVM y el nuevo TCB sean de un tipo que es probable que se utilice en el futuro.

En ciertas circunstancias, podría haber un número excepcionalmente grande de solicitudes de JVM que hayan esperado más tiempo que el periodo crítico. Por ejemplo, esto podría suceder cuando se acaba de realizar un volcado del sistema, lo que retarda todos los procesos. En este caso, en lugar de abandonar la coincidencia y dar a cada solicitud en espera la siguiente JVM disponible, como sucedería normalmente cuando una solicitud ha estado esperando más tiempo que el periodo crítico, CICS aumenta temporalmente el valor del periodo crítico para la agrupación de JVM. Esto le permite realizar coincidencia para las solicitudes en espera y evita producir una sobrecarga anormal. Cuando la situación ha pasado, CICS vuelve a reducir el valor del periodo crítico.

Lanzar y terminar JVM e inhabilitar la agrupación de JVM de forma manual

CICS lanza JVM como respuesta a los requisitos de aplicaciones y reduce automáticamente el número de JVM disponibles si la carga de trabajo no las hace necesarias. También se puede controlar la agrupación de JVM utilizando mandatos de CICS; es posible lanzar y terminar JVM e inhabilitar la agrupación de JVM temporalmente. Este control manual permite implementar cambios en perfiles de JVM o suspender la actividad en la agrupación de JVM. También se puede utilizar para crear JVM antes de las peticiones de aplicación.

Normalmente CICS gestiona el inicio y la terminación de JVM para lograr un nivel equilibrado de capacidad en la agrupación de JVM que permita satisfacer la demanda de las aplicaciones. CICS tiene mecanismos sofisticados para gestionar el número y el tipo de JVM de la agrupación, especialmente cuando es necesario optimizar el rendimiento de cargas de trabajo complejas en momentos de máxima demanda.

Tal vez desee lanzar o terminar JVM de forma manual en algunas situaciones:

- Es necesario actualizar JVM si realiza cambios en perfiles de JVM o en archivos de propiedades de JVM mientras se ejecuta CICS, lo que incluye añadir clases o archivos JAR a vías de acceso de clases.
- Si la carga de trabajo de Java es regular, predecible e implica un número limitado de distintos perfiles de JVM, podría pensar en lanzar JVM de forma manual antes de la demanda de las aplicaciones, de forma que estén listas para su uso tan pronto como sean necesarias.

Lanzamiento de JVM utilizando mandatos de CICS

Para lanzar JVM de forma manual, utilice el mandato **EXEC CICS** o **CEMT PERFORM JVMPOOL**. Es necesario especificar el número de JVM que iniciar, así como el perfil de JVM y la clave de ejecución que se utilizarán para ellas.

El número que se especifique, añadido al número de JVM que ya existen en la agrupación de JVM, no debe superar el límite de **MAXJVMTCBS** para la región CICS. Puede comprobarlo emitiendo el mandato **EXEC CICS** o **CEMT INQUIRE DISPATCHER**. **MAXJVMTCBS** muestra el límite y ACTJVMTCBS muestra el número de JVM que existen en este momento.

CICS no lanza todas las JVM a la vez, sino que planifica los inicios durante un periodo de tiempo breve. Cada JVM está disponible para que la utilice una aplicación en cuanto se ha iniciado. Si una JVM no es utilizada por ninguna aplicación, al igual que cualquier otra JVM desocupada, pasa a ser elegible para terminación en el umbral de tiempo de espera que se haya especificado en el perfil de JVM.

Si acaba de terminar unas JVM para implementar cambios en perfiles de JVM y la actividad de la aplicación en la región CICS es baja, puede utilizar el mandato **PERFORM JVMPOOL** para lanzar una JVM del tipo en el que se aplicaron los cambios. Esto le permite confirmar, sin esperar a una petición de aplicación, que la JVM puede lanzarse con el perfil cambiado y que se pueden cargar las clases especificadas en las vías de acceso de clases.

Si la carga de trabajo de Java en la región CICS es regular y predecible, tal vez desee utilizar el recurso de inicio manual para crear una agrupación de JVM que anticipe las necesidades de las aplicaciones, en lugar de permitir que CICS lo haga en respuesta a la demanda. Esta estrategia puede reducir el tiempo de retardo para aplicaciones en periodos de tiempo en las que aumente la carga de trabajo.

Al configurar el umbral de tiempo de espera (que de forma predeterminada es 30 minutos) y lanzar JVM antes de que sean necesarias, podría estructurar una agrupación de JVM que siempre tenga capacidad disponible para los requisitos. Por ejemplo, podría lanzar un número suficiente de JVM para gestionar las cargas de trabajo pico, con sus umbrales de tiempo de espera definidos de forma que solo sean elegibles para terminación automática tras 24 horas de inactividad. (Tal vez desee definir una tarea que lance el número adecuado de JVM cuando se lance la región CICS). Con una agrupación de JVM como esta, CICS no terminaría las JVM de forma automática en horas del día en las que la carga de trabajo sea reducida. Únicamente se terminarían si el sistema estuviera desocupado durante un periodo prolongado o si la carga de trabajo se redujera a largo plazo.

Cuando se lanzan JVM de forma manual con un perfil de JVM concreto, estas son elegibles para no coincidencia o robo del mismo modo que las JVM lanzadas por CICS. La no coincidencia y el robo cambian el perfil de JVM o la clave de usuario, de forma que la JVM ya no puede ser utilizada por las aplicaciones para las que se lanzó originalmente. La no coincidencia y el robo también implican reiniciar la JVM, lo que puede hacer desaparecer cualquier beneficio de lanzar las JVM por adelantado. La posibilidad de no coincidencia y de robo aumenta a la vez que lo hace el número de perfiles de JVM distintos que haya en la región CICS, por lo que si se desea estructurar una agrupación de JVM de forma manual, el beneficio es probable que sea mayor si las aplicaciones utilizan solo un perfil de JVM o un número pequeño de ellos.

Terminación de JVM

Para terminar JVM, utilice el mandato CEMT o **EXEC CICS PERFORM JVMPOOL**. Puede optar por terminar todas las JVM de la agrupación de JVM o puede especificar un perfil de JVM para terminar solo las JVM con dicho perfil.

Es necesario terminar JVM para implementar cambios en perfiles de JVM o para añadir nuevas clases de aplicación. Los cambios a las clases existentes en la vía de acceso de clase estándar no requieren la terminación de las JVM. La vía de acceso de clase estándar, es la opción recomendada para JVM autónomas, pero si está en el proceso de migrar desde JVM que se restablecen a JVM continuas, en las JVM autónomas tal vez aún tenga clases en la vía de acceso de clases de aplicación compartible.

El mandato **PERFORM JVMPOOL** no termina la memoria caché de clase compartida. La memoria caché de clase compartida se actualiza automáticamente cuando se cambian clases o se añaden nuevas clases, por lo que no es necesario terminarla en esta situación.

Para minimizar las molestias a sus aplicaciones, intente terminar únicamente aquellos perfiles de JVM en los que haya realizado cambios al perfil de JVM, a su archivo de propiedades de JVM o las aplicaciones que lo utilizan. La terminación de un subconjunto de la agrupación de JVM es más eficiente que la terminación de toda la agrupación. Asegúrese de que realmente se terminan todas las JVM afectadas por los cambios. Por ejemplo, una clase Java compartida que haya cambiado puede listarse en la vía de acceso de clases de más de un perfil de JVM. En ciertas circunstancias poco habituales, una clase de aplicación pueden utilizarla JVM con más de un perfil, pero esto tal vez no sea obvio a partir de los perfiles de JVM. Esto puede ser un problema si, por ejemplo, se utilizan cargadores de clases o se crean instancias de clases mediante reflejo, o si se tienen enterprise beans que llaman a otros enterprise beans. Si no se está seguro de si una clase de aplicación la utilizan JVM con más de un perfil, tal vez prefiera asegurarse y terminar toda la agrupación de JVM.

CICS lanza nuevas JVM en cuanto recibe solicitudes de cada tipo de JVM de parte de aplicaciones. Si lo prefiere, puede lanzar JVM de forma manual utilizando el mandato **PERFORM JVMPOOL**. Si ha realizado algún cambio en los perfiles de JVM, las nuevas JVM utilizan las opciones cambiadas. Si ha realizado algún cambio en las aplicaciones Java, las nuevas JVM cargan las clases nuevas o cambiadas.

Inhabilitación de la agrupación de JVM

Para suspender toda actividad en la agrupación de JVM, utilice el mandato **EXEC CICS** o **CEMT SET JVMPOOL** para definir el estado en DISABLED. En este estado, la agrupación de JVM no puede atender nuevas solicitudes.

Cuando se inhabilita la agrupación de JVM, se retienen las JVM que hay en ella, pero los nuevos programas no pueden utilizarlas hasta que se vuelva a habilitar dicha agrupación de JVM. Los programas Java que ya utilizan una JVM tienen permiso para terminar de ejecutarse. Para volver a habilitar la agrupación de JVM, utilice el mandato **EXEC CICS** o **CEMT SET JVMPOOL** para definir el estado en ENABLED.

Inicio de la memoria caché de clase compartida

De forma predeterminada, la memoria caché de clase compartida se inicia automáticamente en cuanto CICS recibe una solicitud para ejecutar una aplicación Java en una JVM en agrupación cuyo perfil requiera utilizar la memoria caché de clase compartida. Si en cualquier momento se detiene la memoria caché de clase compartida y se quiere volver a reiniciar, se puede habilitar el inicio automático o utilizar mandatos de CICS.

Acerca de esta tarea

El parámetro de inicialización del sistema JVMCCSTART controla el comportamiento de inicio normal de la memoria caché de clase compartida. El valor predeterminado es AUTO, con el que la memoria caché de clase compartida se inicia siempre que una JVM en agrupación la necesita. Si hay una memoria caché de clase compartida activa cuando la región CICS se cierra en un inicio en caliente o de emergencia, normalmente esta persiste salvo en algunas situaciones, como una carga inicial de programa (IPL) de z/OS.

Procedimiento

1. Asegúrese de que el valor del parámetro de inicialización del sistema **JVMCCSTART** está definido como AUTO o YES. La memoria caché de clase se inicia cuando la primera JVM en agrupación la necesita.
2. Para reiniciar la memoria caché de clase compartida mientras CICS se ejecuta, utilice uno de los siguientes métodos:
   - Para reiniciar la memoria caché de clase compartida inmediatamente, utilice el mandato **CEMT PERFORM CLASSCACHE START** (o el mandato equivalente de **EXEC CICS**). Si desea habilitar el inicio automático, utilice la opción AUTOSTARTST en el mandato. Puede utilizar la opción CACHESIZE de este mandato si desea cambiar el tamaño de la memoria caché de clase compartida.
   - Para definir que la memoria caché de clase compartida se inicie cuando la necesite una JVM, utilice el mandato **CEMT SET CLASSCACHE AUTOSTARTST** (o el mandato equivalente **EXEC CICS**) para habilitar el inicio automático mientras CICS se esté ejecutando.

Resultados

La memoria caché de clase compartida se reinicia cuando CICS recibe una solicitud para ejecutar una aplicación Java en una JVM en agrupación que requiera la memoria caché de clase compartida. Los posteriores inicios en caliente o de emergencia de CICS utilizan este valor para el inicio automático, a no ser que se haya especificado el parámetro de inicialización del sistema **JVMCCSTART** como sustitución en el inicio.

Ajuste del tamaño de la memoria caché de clase compartida

Cuando se inicia la memoria caché de clase compartida, se fija la cantidad de almacenamiento en la memoria caché. El tamaño predeterminado es 24 MB. Cuando el almacenamiento en la memoria caché de la clase compartida se llena, las clases que ya están en ella se pueden seguir utilizando, pero no se pueden añadir nuevas clases. En esta situación, debe aumentar el tamaño de la memoria caché de clase compartida.

Acerca de esta tarea

CICS proporciona mandatos y parámetros para ayudar a controlar el tamaño de la memoria caché de clase compartida para JVM en agrupación. Si utiliza memorias caché de clase con servidores de JVM, no puede utilizar estos mandatos y debe utilizar el soporte provisto por Java. Para obtener más información sobre el programa de utilidad de clases compartidas de Java, consulte la Java Guía de diagnósticos.

El tamaño de la memoria caché de clase compartida debe ser suficiente para contener todas las clases para sus aplicaciones, como se especifica en la vía de acceso de clase estándar para todas las JVM en agrupación que utilizan la memoria caché de clase compartida. La memoria caché de clase compartida no distingue entre clases de aplicación que se pueden compartir y que no y no contiene código compilado por JIT.

Procedimiento

1. Calcule el almacenamiento necesario para las clases de aplicación o, para lograr mejores resultados, ejecute las aplicaciones en un entorno de prueba para identificar el espacio total necesario en la memoria caché de clase compartida:

a. Ejecute cada aplicación de forma repetida en un entorno de prueba utilizando la memoria caché de clase compartida.
b. Mientras ejecute la aplicación, supervise la cantidad de espacio libre en la memoria caché de clase compartida. Utilice el mandato **INQUIRE CLASSCACHE** para informar sobre el tamaño de la memoria caché de clase compartida y la cantidad de almacenamiento libre de dicha memoria caché especificando las opciones CACHESIZE y CACHEFREE.

   Puede obtener más estadísticas de la memoria caché de clase compartida ejecutando el siguiente mandato en un shell de z/OS UNIX System Services:

   ```
   java -Xshareclasses:name=CICS_sharedcc_APPLID_n,printStats
   ```

   donde *APPLID* es el identificador de aplicación de z/OS Communications Server del sistema CICS y *n* es el número de generación actual para la memoria caché de clase compartida.
c. Ejecute la aplicación hasta que la cantidad de espacio libre se haya estabilizado.
d. Repita este proceso para cada aplicación que utilice la memoria caché de clase compartida.
e. Añada la cantidad de almacenamiento utilizado por cada aplicación y sume un margen de seguridad adecuado para cualquier cambio futuro en las aplicaciones.

   El total le brinda un tamaño aproximado para la memoria caché de clase compartida.
2. Utilice el mandato **PERFORM CLASSCACHE RELOAD** para crear una nueva memoria caché de clase compartida. Puede especificar el tamaño para la nueva memoria caché de clase compartida mediante la opción CACHESIZE del mandato. Este mandato es el que menos altera las JVM en agrupación que utilizan la memoria caché de clase compartida.
3. Opcional: Cambie el valor para el parámetro de inicialización del sistema JVMCCSIZE. Este parámetro especifica el tamaño inicial de la memoria caché de clase compartida y se utiliza en reinicios en frío e iniciales de CICS.

Resultados

Cuando se especifica un nuevo tamaño para la memoria caché de clase compartida con CICS en ejecución, los siguientes reinicios en caliente y de emergencia de CICS utilizan el nuevo valor. Los reinicios iniciales y en frío de CICS utilizan el valor del parámetro de inicialización del sistema **JVMCCSIZE**.

Terminación de la memoria caché de clase compartida

Puede utilizar CICS para terminar la memoria caché de clase que utilizan JVM en agrupación e impedir que se reinicie. También puede terminar cualquier JVM en agrupación que la utilice.

Acerca de esta tarea

Cuando se termina la memoria caché de clase compartida y se habilita el inicio automático, se crea una nueva memoria caché de clase compartida en cuanto una JVM en agrupación solicita su uso. Si desea terminar la memoria caché de clase compartida sin reiniciarla, debe inhabilitar el inicio automático.

Si termina la memoria caché de clase compartida y no se reinicia, las JVM en agrupación que utilizan dicha memoria no se pueden ejecutar.

Cuando se cambia el estado del inicio automático de la memoria caché de clase compartida con CICS en ejecución, los siguientes reinicios en caliente de CICS utilizan el valor más reciente definido. Si la región CICS se inicia como INITIAL o COLD, o si se especifica el parámetro de inicialización del sistema **JVMCCSTART** como una sustitución en el inicio, se utiliza el valor del parámetro de inicialización del sistema.

Procedimiento

1. Compruebe el estado del inicio automático en la memoria caché de la clase compartida. Puede utilizar la vista de operaciones de la memoria caché de clase de JVM en CICS Explorer o el mandato **INQUIRE CLASSCACHE**.
2. Si no desea que la memoria caché de clase compartida se reinicie tras terminarla, inhabilite el inicio automático. Puede inhabilitar el inicio automático para la memoria caché de clase compartida de tres formas distintas:
   - Antes de especificar el mandato para terminar la memoria caché de clase compartida, utilice el mandato **SET CLASSCACHE AUTOSTARTST** para inhabilitar el inicio automático.
   - Cuando especifique el mandato **PERFORM CLASSCACHE** para terminar la memoria caché de clase compartida, utilice la opción AUTOSTARTST para inhabilitar el inicio automático.
   - Para inhabilitar el inicio automático en la siguiente ejecución de CICS, defina el parámetro de inicialización del sistema **JVMCCSTART** en NO. Este valor siempre impide el inicio automático en un inicio en frío o inicial de CICS. Si la memoria caché de clase compartida está activa cuando se cierra la región, esta persiste en un inicio en caliente o de emergencia, incluso si se especifica **JVMCCSTART** como sustitución.
3. Termine la memoria caché de clase compartida y cualquier JVM en agrupación que la utilice. Puede utilizar la vista de operaciones de la memoria caché de clase de JVM o el mandato **PERFORM CLASSCACHE**. Puede depurar o forzar la depuración de las JVM, o dejarlas que terminen de ejecutar sus programas Java actuales antes de suprimirlas. Las JVM que no estén utilizando la memoria caché de clase compartida no se ven afectadas por este mandato.
4. Repita el mandato **PERFORM CLASSCACHE** para intentar depurar las tareas que estén utilizando las JVM en agrupación, si no desea reiniciar la memoria caché de clase compartida y las JVM en agrupación que la usan permanecen activas demasiado tiempo. Repita el mandato únicamente si el inicio automático para la memoria caché de clase compartida está inhabilitado. El mandato opera tanto en la memoria caché de clase compartida más reciente como en cualquier memoria caché de clase compartida antigua de la región que todavía esté siendo utilizada por alguna JVM. Si el inicio automático está habilitado y se repite el mandato para terminar la memoria caché de clase compartida, el mandato puede terminar la nueva memoria caché de clase compartida que haya iniciado el recurso de inicio automático.

Resultados

La memoria caché de clase para las JVM en agrupación se termina correctamente.

Supervisión de la memoria caché de clase compartida

Puede utilizar mandatos de CICS para informar sobre el estado de la memoria caché de clase compartida para las JVM en agrupación y para cada JVM de la agrupación.

Procedimiento

- Para informar sobre el estado de la memoria caché de clase compartida para JVM en agrupación, utilice el mandato **CEMT INQUIRE CLASSCACHE** (o el mandato equivalente de **EXEC CICS**). El mandato le indica la clase compartida que se inicializa (STARTING), que está lista para utilizar (STARTED), que se vuelve a cargar (RELOADING) o que no está activa (STOPPED). El mandato también le proporciona información como el estado del inicio automático, el tamaño de la memoria caché de clase compartida y la cantidad de espacio libre en la memoria caché. El mandato también informa de cualquier memoria caché de clase compartida antigua en la región CICS que se esté desactivando gradualmente.
- Para informar sobre el estado de la JVM en la agrupación de JVM, utilice el mandato **CEMT INQUIRE JVM** (o el mandato equivalente de **EXEC CICS**). El mandato le proporciona información sobre una JVM especificada o sobre cada JVM de la agrupación, indicando la tarea a la que está asignado, si su clave de ejecución es USER o CICS y si utiliza la memoria caché de clase compartida o no.

Supervisión de la agrupación de JVM

Puede utilizar el mandato **CEMT INQUIRE JVMPOOL** (o el mandato equivalente de **EXEC CICS**) para averiguar información sobre la agrupación de JVM.

El mandato le informa sobre lo siguiente:

- El número de JVM en la agrupación.
- El número de dichas JVM que se ha marcado para su supresión pero que todavía está utilizando una tarea.
- Si la agrupación de JVM está habilitada o inhabilitada (es decir, si puede atender nuevas solicitudes o no).
- Qué opciones de rastreo se aplican para las JVM de la agrupación (esta opción solo está disponible en la versión de **EXEC CICS** del mandato).

Supervisión de las JVM de la agrupación de JVM

Puede utilizar el mandato **EXEC CICS INQUIRE JVM** o el mandato **CEMT INQUIRE JVM** para identificar el estado de cada JVM de la agrupación de JVM e informar de él. También puede supervisar la actividad en la agrupación de JVM mediante las estadísticas de CICS.

Mediante el mandato **EXEC CICS INQUIRE JVM**, puede consultar sobre una JVM específica o puede examinar todas las JVM de la agrupación de JVM. Mediante el mandato CEMT INQUIRE JVM, puede listar todas las JVM de la agrupación de JVM o consultar sobre todas las JVM que se encuentren en un estado especificado.

Los mandatos le informan sobre lo siguiente:

- El perfil de JVM y la clave de ejecución de las JVM de la agrupación.
- Qué JVM de la agrupación utilizan la memoria caché de clase compartida.
- Cuánto tiempo tiene cada JVM.
- La tarea a la que está asignada una JVM y la hora a la que se ha asignado a dicha tarea.

- Las JVM que se están desactivando gradualmente como resultado de un mandato **CEMT SET JVMPOOL PHASEOUT**, **PURGE** o **FORCEPURGE**, o de un mandato **CEMT PERFORM CLASSCACHE PHASEOUT**, **PURGE** o **FORCEPURGE** (o los mandatos de **EXEC CICS** equivalentes).

También puede supervisar la actividad en la agrupación de JVM mediante las estadísticas de CICS. Utilice el mandato **EXEC CICS COLLECT STATISTICS** o el mandato **CEMT PERFORM STATISTICS**, con las opciones correspondientes, para recopilar estas estadísticas. Algunas estadísticas útiles son las estadísticas de agrupación de JVM (opción JVMPOOL), las estadísticas de la modalidad de TCB (opción DISPATCHER), las estadísticas del perfil de JVM (opción JVMPROFILE) y las estadísticas del programa de JVM (opción JVMPROGRAM). Estas estadísticas pueden indicarle, entre otras cosas:

- Cuántas JVM de un perfil concreto, en una modalidad de TCB concreta, hay en la agrupación de JVM (a partir de las estadísticas del perfil de JVM).
- Cuántas solicitudes se realizaron de una JVM de un perfil concreto, en una modalidad de TCB concreta (a partir de las estadísticas del perfil de JVM).
- Cuántas veces tuvo que esperar una solicitud de una JVM porque no había ninguna JVM disponible con una clave de ejecución y un perfil que coincidieran con la solicitud (a partir de las estadísticas de la agrupación de TCB para la agrupación de JVM). Esto incluye tanto solicitudes a las que se asignara finalmente una JVM adecuada y solicitudes a las que CICS decidiera asignar una JVM no coincidente o robada en lugar de hacerlas esperar más tiempo. Esta cifra también puede incluir esperas de serialización; es decir, el tiempo pasado esperando para obtener cualquier bloqueo necesario.
- Cuánto tiempo esperaron estas una solicitudes (a partir de las estadísticas de la agrupación de TCB para la agrupación de JVM).
- Cuántas veces se asignó a una solicitud de JVM una JVM que tuviera el perfil incorrecto o la clave de ejecución equivocada (a partir de las estadísticas de perfil de JVM). Estos incidentes de no coincidencia y de robo se desglosan por perfil de JVM, por lo que puede ver si un perfil concreto produce un exceso de actividad de robo.

Supervisión del uso de perfiles de JVM en agrupación

Puede utilizar el mandato **EXEC CICS INQUIRE JVMPROFILE** en modalidad de examen para encontrar qué perfiles de JVM se han utilizado para JVM en agrupación desde que se inició la región CICS. También puede recopilar estadísticas de CICS para perfiles de JVM.

INQUIRE JVMPROFILE encuentra perfiles de JVM que se hayan utilizado durante el tiempo de vida de la región CICS. El mandato devuelve el nombre de perfil de cada JVM, tal y como se utiliza en un recurso PROGRAM, y el nombre completo de vía de acceso del archivo z/OS UNIX correspondiente a ese perfil de JVM. El mandato también le indica si hay alguna JVM con ese perfil que utilice la memoria caché de clase compartida.

Puede recopilar estadísticas para perfiles de JVM utilizando el mandato **EXEC CICS COLLECT STATISTICS** o el mandato **CEMT PERFORM STATISTICS**. Para ambos mandatos, especifique la opción JVMPROFILE. Las estadísticas se desglosan por perfil de JVM y por clave de ejecución, y muestran, entre otras cosas:

- El número de solicitudes de JVM de este perfil realizadas por aplicaciones.
- El número total, actual y pico de las JVM de este perfil que estaban en la agrupación de JVM.

- El número de JVM en agrupación de este perfil que se destruyeron debido a que CICS tenía poco espacio de almacenamiento.
- La incidencia de robos de TCB por las JVM de este perfil y desde estas.
- El almacenamiento dinámico de Language Environment y el almacenamiento dinámico de JVM utilizados por las JVM de este perfil.

Servidor JVM y estadísticas JVM agrupadas en la Guía de rendimiento contiene más información sobre estadísticas de JVM e indica cómo encontrar los listados y los informes completos correspondientes a estas estadísticas.

Supervisión de programas en JVM en agrupación

Puede utilizar el mandato **EXEC CICS COLLECT STATISTICS** o el mandato **CEMT PERFORM STATISTICS** para recopilar estadísticas sobre programas Java que se ejecuten en JVM en agrupación. Para ambos mandatos, especifique la opción JVMPROGRAM.

CICS no recopila estadísticas para estos programas cuando se emite un mandato **COLLECT** o **PERFORM STATISTICS PROGRAM**, ya que los programas de JVM no los carga CICS.

Para cada programa, las estadísticas muestran los siguientes detalles:

- El perfil de JVM que necesita el programa, según se especifica en el atributo JVMPROFILE del recurso PROGRAM.
- La clave de ejecución que necesita el programa, bien clave de CICS o bien clave de usuario, según se especifica en el atributo EXECKEY del recurso PROGRAM.
- La clase principal del programa, según se especifica en el atributo JVMCLASS del recurso PROGRAM.
- El número de veces que se ha utilizado el programa.

Para obtener más información sobre estadísticas de JVM, consulte Servidor JVM y estadísticas JVM agrupadas en la Guía de rendimiento.

Utilización de DFHJVMAT para modificar opciones en un perfil de JVM

DFHJVMAT es un programa sustituible por el usuario que se puede utilizar para sustituir las opciones especificadas en un perfil de JVM para JVM en agrupación de un solo uso. Normalmente, un perfil de JVM proporciona flexibilidad suficiente como para configurar una JVM como sea necesario. Utilice DFHJVMAT únicamente si debe personalizar la JVM de un modo que no se pueda conseguir especificando opciones en el perfil de JVM.

También se puede utilizar DFHJVMAT para sustituir el atributo JVMCLASS en un recurso PROGRAM de CICS. Este atributo especifica la clase principal en el programa Java que se va a ejecutar en la JVM. Si se utiliza el recurso PROGRAM, el límite para el atributo JVMCLASS es de 255 caracteres, pero puede utilizar DFHJVMAT para especificar un nombre de clase superior a 255 caracteres.

Puede llamar a DFHJVMAT especificando INVOKE_DFHJVMAT=YES como una opción en el perfil de JVM que se desea sustituir.

Importante

Solo puede llamar a DFHJVMAT para JVM en agrupación de un solo uso; es decir, una JVM con un perfil de JVM que especifique la opción REUSE=NO. Con las JVM

de un solo uso, cuando la tarea que utiliza la JVM termina, CICS no intenta que la JVM esté disponible para su reutilización por parte de otra tarea.

No puede llamar a DFHJVMAT para JVM en agrupación continuas ni para un servidor de JVM. Si especifica INVOKE_DFHJVMAT=YES para cualquier tipo de JVM, INVOKE_DFHJVMAT=YES se ignora y no se llama a DFHJVMAT.

Los valores especificados en el perfil de JVM están disponibles para DFHJVMAT como variables de entorno de z/OS UNIX System Services, que se pueden modificar antes de crear la JVM.

Nota: Los valores de los parámetros STDERR y STDOUT, que pueden ser interpretados por CICS para generar nombres específicos de tarea, se pasan a DFHJVMAT antes de la interpretación.

DFHJVMAT utiliza las funciones `getenv` y `setenv` de C/C++ para modificar las variables de entorno que se correspondan con las opciones del perfil de JVM. Por ejemplo, puede utilizar el siguiente mandato para sustituir la variable de entorno CLASSPATH_SUFFIX por el valor especificado:

```
setenv(ecp_suffix, cp_suffix_val,1)
```

donde:

```
char *ecp_suffix = "CLASSPATH_SUFFIX";
char *cp_suffix_val ="/u/jtest1/Java/test:.";
```

La función `setenv` no tiene ningún efecto en el recurso PROGRAM de CICS y tiene vigencia solo durante el tiempo de vida de la JVM.

DFHJVMAT proporcionado por CICS:

- Emite solicitudes de `getenv` para cada variable.
- Emite una `printf` para la `stdout` de destino, con el fin de registrar el valor de cada variable.
- Contiene (dentro de los comentarios) código de ejemplo que enseña cómo utilizar el mandato `setenv` para los nombres provistos para **stdout** y **stderr**, con el fin de crear archivos exclusivos de salida y de error para cada tarea de CICS.

Si escribe su propio programa para personalizar opciones del perfil de JVM basado en la versión provista, el nombre tiene que ser DFHJVMAT y el programa debe escribirse en C. Puede utilizar mandatos de EXEC CICS en DFHJVMAT, pero dichos mandatos pueden aumentar el tiempo de proceso. DFHJVMAT debe escribirse en base a estándares de enhebramiento seguro y definirse con CONCURRENCY(THREADSAFE) en su recurso PROGRAM, ya que pueden ejecutarse varias invocaciones de este módulo en paralelo.

Opciones del perfil de JVM que están disponibles para DFHJVMAT

Las opciones del perfil de JVM indicadas en este tema son disponibles para DFHJVMAT. Las opciones se leen desde el perfil de JVM especificado y se crean como variables de entorno utilizando los servicios de Language Environment.

En la mayoría de los casos, las descripciones completas de estas opciones se encuentran en “Perfiles de JVM: opciones y ejemplos” en la página 104. Antes de modificar alguna de estas opciones utilizando DFHJVMAT, lea la descripción completa de la opción.

Algunas de las opciones ya no se describen en la documentación de CICS, y la información acerca de estas se puede encontrar en la documentación para IBM 64 bits SDK para z/OS, Java Technology Edition y en otra documentación de Java.

Nota:

1. Excepto cuando se indique explícitamente que es solo por motivos de información, puede restablecer los valores de estas variables.
2. Todas las variables de entorno y los valores distinguen entre mayúsculas y minúsculas y deben definirse como se muestra.
3. CICS ignora cualquier valor que no reconoce como una opción válida.
4. En el caso de las opciones que comiencen por `X`:
   - Algunas de las opciones ya no se describen en la documentación de CICS. Sin embargo, siguen siendo opciones válidas y están disponibles para DFHJVMAT.
   - Ahora estas opciones deben comenzar por `-X` cuando se especifique en un perfil de JVM. Sin embargo, el guión no se incluye en las variables de entorno utilizadas por DFHJVMAT, que aún comienzan por `X`.

Tabla 11. Opciones de perfil de JVM disponibles para DFHJVMAT

Opción	**Especifica**
CLASSPATH_PREFIX, CLASSPATH_SUFFIX	Prefijo y sufijo para vía de acceso de clase estándar
INVOKE_DFHJVMAT	Solo para fines de información.
JAVA_DUMP_OPTS	Conjunto de opciones de volcado de Java para obtener diagnóstico en caso de una terminación anómala en la JVM.
JAVA_HOME	Vía de acceso a subdirectorios y archivos JAR de IBM 64 bits SDK para z/OS, Java Technology Edition
JVMPROPS	Vía de acceso y nombre del archivo de propiedades de la JVM.
LIBPATH_PREFIX, LIBPATH_SUFFIX	Prefijo y sufijo para vía de acceso a biblioteca.
STDERR	Nombre del archivo z/OS UNIX para la salida **stderr** desde la JVM.
STDIN	Nombre del archivo z/OS UNIX para **stdin**.
STDOUT	Nombre del archivo z/OS UNIX para la salida **stdout** desde la JVM.
USEROUTPUTCLASS	Nombre de una clase Java que intercepta y redirige las salidas **stdout** y **stderr** desde la JVM.
VERBOSE	Nivel de mensajes informativos desde la JVM
WORK_DIR	Directorio de trabajo para la región CICS en z/OS UNIX
`Xcheck`	Realice comprobaciones adicionales.
`Xdebug`	Habilite el soporte de depuración.
`Xmaxe`, `Xmaxf`, `Xmine`, `Xminf`	Tamaños máximo y mínimo de expansión de almacenamiento dinámico y tamaños de porcentaje libre de almacenamiento dinámico para el almacenamiento dinámico.
`Xms`	Tamaño inicial del almacenamiento dinámico.
`Xmx`	Tamaño máximo del almacenamiento dinámico.

Tabla 11. Opciones de perfil de JVM disponibles para DFHJVMAT (continuación)

Opción	Especifica
`Xnoagent`	Inhabilita el agente `sun.tools.debug` antiguo (si se ha especificado `Xdebug`).
`Xnoclassgc`	Inhabilita la recogida de basura de clase.
`Xoss`	Tamaño máximo de pila de Java para cualquier hebra.
`Xrs`	Reduce el uso de señales del sistema operativo por parte la JVM.
`Xrun`*`nombre_dll`*	Carga la biblioteca de enlace dinámico (DLL) especificada y la pasa a las opciones especificadas.
`Xss`	Tamaño de pila para cada nueva hebra de Java.
`Xverify`	Nivel de verificación que realizar en las clases cargadas.

Dos campos adicionales que no se encuentran en un perfil de JVM estándar se pasan a DFHJVMAT, de la siguiente forma:

CICS_PROGRAM
Especifica el nombre del programa de CICS (1-8 caracteres) asociado con la clase Java que ejecutar. Este nombre se establece en el tiempo de ejecución; se pasa a DFHJVMAT con fines informativos únicamente y no se puede cambiar. CICS ignora cualquier cambio.

CICS_PROGRAM_CLASS
Especifica el nombre de clase de la aplicación de usuario de CICSy se obtiene de la definición de recurso del programa. Esta se define mediante el atributo JVMCLASS en la definición de recurso PROGRAM de CICS. Como alternativa a utilizar DFHJVMAT para sustituir este atributo, puede utilizar el mandato **SET PROGRAM** para modificar el atributo JVMCLASS en el recurso PROGRAM antes de pasar el control a la JVM. Si se utiliza el recurso PROGRAM, el límite para el atributo JVMCLASS es de 255 caracteres, pero puede utilizar DFHJVMAT para especificar un nombre de clase superior a 255 caracteres.

Capítulo 7. Mejora del rendimiento de Java

Puede realizar distintas acciones para mejorar el rendimiento tanto de las aplicaciones Java como de las JVM en las que se ejecutan.

Acerca de esta tarea

No importa lo bien que se haya ajustado CICS: si una aplicación está escrita de forma ineficaz, siempre tendrá un rendimiento inferior al de aplicaciones bien escritas. Por ejemplo, si cambia las aplicaciones para que generen menos basura, puede lograr ahorros importantes en los costes de la recogida de basura. Si se produce menos basura, se emplea menos tiempo en la recogida de basura. Para mejorar el rendimiento, asegúrese siempre de que las aplicaciones Java se escriben de forma eficaz y de que se ajusta el entorno Java.

Procedimiento

1. Determine los objetivos de rendimiento para la carga de trabajo de Java. Algunos de los objetivos más comunes incluyen minimizar el uso del procesador o los tiempos de respuesta de la aplicación. Tras haber decidido el objetivo, puede ajustar el entorno Java adecuadamente.
2. Analice las aplicaciones Java para asegurarse de que se ejecutan de forma eficiente y no generan demasiada basura. IBM dispone de herramientas que pueden ayudarle a analizar las aplicaciones Java para mejorar la eficacia y el rendimiento tanto de métodos concretos como de la aplicación en general.
3. Ajuste el servidor de JVM o las JVM en agrupación. Puede utilizar estadísticas y herramientas de IBM para analizar los valores de almacenamiento, la recogida de basura, las esperas de las tareas y otra información con el fin de ajustar el rendimiento de la JVM.
4. Ajuste el enclave de Language Environment en el que se ejecuta una JVM. Las JVM utilizan almacenamiento de MVS, que se obtiene mediante llamadas a servicios de MVS de Language Environment. Puede modificar las opciones de tiempo de ejecución para Language Environment con el fin de ajustar el almacenamiento que asigna MVS.
5. Opcional: Si utiliza la región de biblioteca compartida de z/OS para compartir DLL entre distintas JVM en diferentes regiones CICS, puede ajustar los valores de almacenamiento.

Determinación de los objetivos de rendimiento para la carga de trabajo de Java

Ajustar las JVM de CICS para lograr el mejor rendimiento general para la carga de trabajo de una aplicación determinada implica distintos factores. Debe decidir cuáles son las característica de rendimiento que desea para su carga de trabajo de Java. Cuando establezca estas características, puede determinar qué parámetros cambiar y cómo cambiarlos.

Los siguientes objetivos de rendimiento para cargas de trabajo de Java son los más comunes:

Uso mínimo del procesador general
Este objetivo prioriza el uso más eficiente del recurso de procesador disponible. Si se ajusta una carga de trabajo para lograr este objetivo, el

uso total del procesador en toda la carga de trabajo se minimiza, pero las tareas individuales pueden experimentar un alto consumo de procesador. El ajuste para el uso mínimo de procesador implica especificar tamaños de almacenamiento dinámico grandes para las JVM, con el fin de minimizar el número de recogidas de basura.

Tiempos mínimos de respuesta de las aplicaciones
Este objetivo prioriza el garantizar que una tarea de aplicación se devuelva al interlocutor lo más rápido posible. Este objetivo puede ser especialmente importante si han de lograrse acuerdos de nivel de servicio. Si una carga de trabajo se ajusta para lograr este objetivo, las aplicaciones responden de forma rápida y consistente, aunque puede producirse un uso de procesador mayor para recogidas de basura. El ajuste para tiempos mínimos de respuesta de las aplicaciones implica mantener el tamaño de almacenamiento dinámico pequeño y posiblemente utilizar la política de recogida de basura gencon.

Tamaño de almacenamiento dinámico mínimo para almacenamiento de JVM
Este objetivo prioriza la reducción de la cantidad de almacenamiento que utilizan las JVM. Las JVM utilizan almacenamiento de 64 bits para que sea posible ejecutar muchas JVM en agrupación y muchos servidores de JVM en una región CICS. Si las JVM en agrupación utilizan un almacenamiento dinámico más pequeño, puede que sea posible ejecutar más de ellas en la región CICS. Sin embargo, elegir este objetivo puede aumentar los costes de procesador. Ajustar las JVM para minimizar el tamaño de almacenamiento dinámico resulta en una mayor frecuencia de los sucesos de recogida de basura.

Hay otros factores que pueden afectar a los tiempos de respuesta de las aplicaciones. El más importante es el compilador Just In Time (JIT). El compilador JIT optimiza el código de aplicación de forma dinámica en tiempo de ejecución y brinda muchos beneficios, pero necesita una cierta cantidad de recurso de procesador para hacerlo.

Análisis de aplicaciones Java utilizando el IBM Health Center

Para mejorar el rendimiento de una aplicación Java, puede utilizar el IBM Health Center para analizar la aplicación. Esta herramienta brinda recomendaciones para ayudarle a mejorar el rendimiento y la eficiencia de su aplicación.

Acerca de esta tarea

IBM Health Center está disponible en el IBM Support Assistant Workbench. Estas herramientas gratuitas están disponibles para su descarga de IBM como se describe en la guía Getting Started. Intente ejecutar la aplicación en una JVM propia. Si ejecuta una carga de trabajo mezclada en un servidor de JVM, puede resultar más difícil analizar una aplicación en concreto.

Procedimiento

1. Añada las opciones de conexión necesarias al perfil JVM del servidor de JVM. La documentación de IBM Health Center describe las opciones que es necesario añadir para conectarse a la máquina virtual Java desde la herramienta.
2. Lance IBM Health Center y conéctelo a la JVM en ejecución. IBM Health Center informa de la actividad de la JVM en tiempo real, así que espere unos instantes a que comience a supervisar la JVM.

3. Seleccione el enlace **Profiling** (creación de perfiles) para crear el perfil de la aplicación. Puede comprobar el tiempo dedicado a distintos métodos. Compruebe qué métodos tienen un uso más alto para buscar problemas potenciales.

   Consejo: El separador **Analysis and Recommendations** (análisis y recomendaciones) puede identificar métodos concretos que serían buenos candidatos para optimización.
4. Seleccione el enlace **Locking** (bloqueo) para buscar contenciones de bloqueo en la aplicación. Si la carga de trabajo de Java no puede utilizar todo el procesador disponible, la causa puede ser el bloqueo. El bloqueo en la aplicación puede reducir la cantidad de hebras paralelas que se pueden ejecutar.
5. Seleccione el enlace **Garbage Collection** (recogida de basura) para comprobar el uso de almacenamiento dinámico y recogida de basura. El separador **Garbage Collection** (recogida de basura) le puede indicar cuánto almacenamiento dinámico se utiliza y con qué frecuencia la JVM se detiene para realizar recogida de basura.
   a. Compruebe la proporción de tiempo dedicada a la recogida de basura. Esta información se presenta en la sección Summary (resumen). Si el tiempo dedicado a la recogida de basura es superior al 2%, tal vez deba ajustar dicha recogida de basura.
   b. Compruebe el tiempo de detención para recogida de basura. Si el tiempo de detención es superior a 10 milisegundos, la recogida de basura puede estar afectando a los tiempos de respuesta de la aplicación.
   c. Divida la tasa de recogida de basura por el número de transacciones para averiguar cuánta basura produce cada transacción. Si la cantidad de basura parece alta para la aplicación, tal vez deba investigar esta algo más.

Qué hacer a continuación

Tras haber analizado la aplicación, puede ajustar el entorno Java para su carga de trabajo de Java.

Recogida de basura y expansión de almacenamiento

La recogida de basura y la expansión de almacenamiento dinámico son parte esencial de la operación de una JVM. La frecuencia de la recogida de basura en una JVM se ve afectada por la cantidad de basura o de objetos creados por las aplicaciones que se ejecutan en la JVM.

Anomalías de asignación

Cuando una JVM se queda sin espacio en el almacenamiento dinámico y no puede asignar más objetos (una anomalía de asignación), se desencadena una recogida de basura. El colector de basura limpia los objetos del almacenamiento dinámico a los que ya no hacen referencia las aplicaciones y libera parte del espacio. La recogida de basura detiene la ejecución de todos los demás procesos en la JVM durante la duración del ciclo de recogida de basura, por lo que el tiempo dedicado a la recogida de basura es tiempo que no se utiliza para ejecutar aplicaciones. La Java Guía de diagnósticos contiene una explicación detallada del proceso de recogida de basura de la JVM.

Cuando se desencadena una recogida de basura debido a una anomalía de asignación, pero la recogida de basura no libera suficiente espacio, el colector de basura expande el almacenamiento dinámico. Durante la expansión del

almacenamiento dinámico, el colector de basura toma almacenamiento de la cantidad máxima de almacenamiento reservada para el almacenamiento dinámico (la cantidad especificada por la opción `-Xmx`) y la añade a la parte activa del almacenamiento dinámico (que comenzó como el tamaño especificado por la opción `-Xms`). La expansión del almacenamiento dinámico no aumenta la cantidad de almacenamiento necesario para la JVM, porque la cantidad máxima de almacenamiento especificada por la opción `-Xmx` ya se ha asignado a la JVM en el inicio. Si el valor de la opción `-Xms` proporciona suficiente almacenamiento en la parte activa del almacenamiento dinámico para las aplicaciones, el colector de basura no tiene que realizar ningún tipo de expansión del almacenamiento dinámico.

En algún momento durante el tiempo de vida de la JVM, el colector de basura deja de expandir el almacenamiento dinámico, porque este ha alcanzado un estado en el que el colector de basura está satisfecho con la frecuencia de la recogida de basura y la cantidad de espacio liberado por el proceso. El colector de basura no pretende eliminar las anomalías de asignación, por lo que aún puede desencadenarse alguna recogida de basura por las anomalías de asignación después de que el colector de basura haya detenido la expansión del almacenamiento dinámico. En función de los objetivos de rendimiento, tal vez considere que esta frecuencia de recogida de basura sea excesiva.

Opciones de recogida de basura

Puede utilizar distintas políticas para la recogida de basura que equilibren el rendimiento de la aplicación y el sistema general y los tiempos de pausa que provoca la recogida de basura. La recogida de basura se controla mediante la opción `-Xgcpolicy`:

`-Xgcpolicy:optthruput`
: Esta política proporciona un rendimiento alto para las aplicaciones, pero a costa de pausas ocasionales, cuando se produce la recogida de basura.

`-Xgcpolicy:gencon`
: Esta política ayuda a minimizar el tiempo que se produce en cada pausa de recogida de basura. Utilice esta política de recogida de basura con servidores de JVM. Puede comprobar qué política utiliza el servidor de JVM consultando el recurso JVMSERVER. Las estadísticas del servidor de JVM tienen campos que indican cuántos eventos importantes y menores de recogida de basura se producen y qué tiempo de procesador se dedica a dicha recogida.

Puede modificar la política de recogida de basura actualizando el perfil de JVM. Para obtener detalles sobre todas las opciones de recogida de basura, consulte Specifying garbage collection policy.

Ejemplo 1: una aplicación que produce pequeñas cantidades de basura

La Figura 7 en la página 165 muestra el almacenamiento dinámico de una JVM en agrupación para la política de recogida de basura `optthruput`. La cantidad máxima de almacenamiento reservado para el almacenamiento dinámico la determina la opción `-Xmx`. La parte activa del almacenamiento dinámico, determinada por la opción `-Xms`, se muestra sombreada.

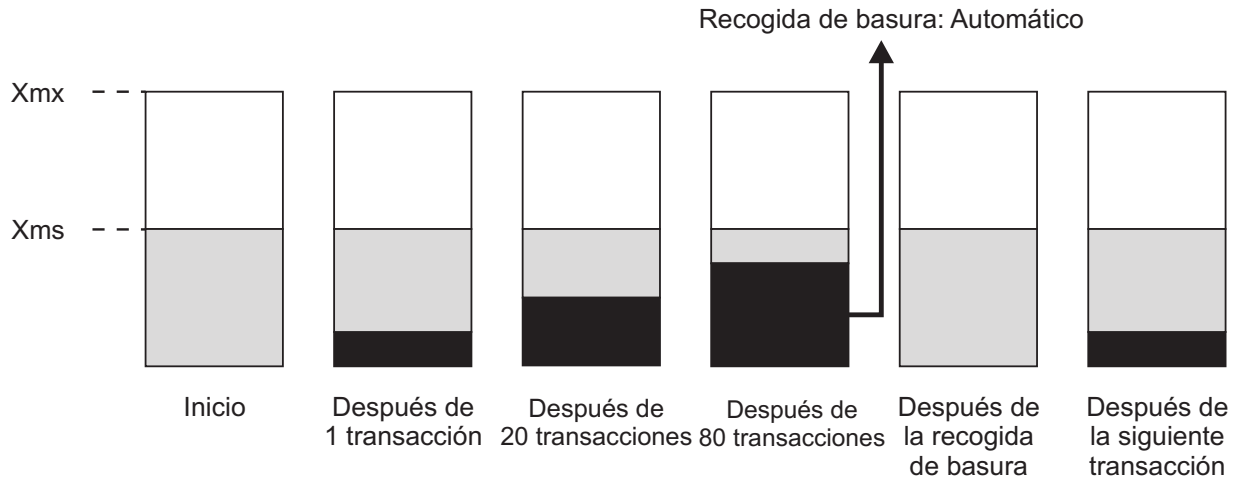


Figura 7. Almacenamiento dinámico de una JVM en agrupación con pequeñas cantidades de basura

En el inicio, `-Xms` se define a la mitad de `-Xmx`, como sucede con los valores predeterminados de los perfiles de JVM proporcionados). El límite de utilización de almacenamiento dinámico (opción `GC_HEAP_THRESHOLD`) se define al valor predeterminado de 85%.

La primera aplicación que se ejecuta en la JVM utiliza una cantidad pequeña del almacenamiento en la parte activa del almacenamiento dinámico. El almacenamiento que utiliza se muestra en negro. Cuando la transacción ha finalizado, ya no se hace referencia a los objetos utilizados por la aplicación, por lo que son elegibles para recogida de basura. Permanecen en el almacenamiento dinámico hasta que se produce la recogida de basura.

Cuando 20 transacciones han utilizado la JVM, la cantidad de almacenamiento ocupado en la parte activa del almacenamiento dinámico ha aumentado. Cada transacción ha utilizado una pequeña cantidad de almacenamiento y aún no se ha producido ninguna recogida de basura.

Tras 80 transacciones, se ha alcanzado el límite del 85% de utilización del almacenamiento dinámico, con el 85% del almacenamiento ocupado en la parte activa del almacenamiento dinámico. Inmediatamente tras la transacción durante la cuál se ha alcanzado el límite, CICS inicia una recogida de basura. Tras la recogida de basura, todos los objetos utilizados por las primeras 80 transacciones se han sometido a recogida de basura, por lo que la parte activa del almacenamiento dinámico ahora está vacía. La siguiente aplicación que se ejecuta en la JVM vuelve a utilizar una cantidad pequeña del almacenamiento, y el ciclo comienza otra vez.

En este ejemplo, no hay anomalías de asignación y no se produce expansión del almacenamiento dinámico, porque el valor de la opción `-Xms` está definido de forma que en la parte activa del almacenamiento dinámico hay suficiente almacenamiento para las aplicaciones. Solo se producen las recogidas de basura solicitadas por CICS en el límite de utilización del almacenamiento dinámico. Sin embargo, si asumimos que esta carga de trabajo permanece constante, la opción `-Xmx` es más alta de lo necesario. Todo el almacenamiento reservado para el almacenamiento dinámico se ha asignado, pero no se utiliza y, por tanto, se malgasta.

Ejemplo 2: una aplicación multihebra que produce grandes cantidades de basura

La Figura 8 muestra el almacenamiento dinámico de un servidor JVM en varias etapas para la política de recogida de basura `gencon` . A diferencia de una JVM en agrupación, un servidor de JVM puede ejecutar varias solicitudes para una aplicación al mismo tiempo. Por lo tanto, la aplicación puede producir cantidades más grandes de basura. En un servidor de JVM, esta maneja automáticamente la recogida de basura.

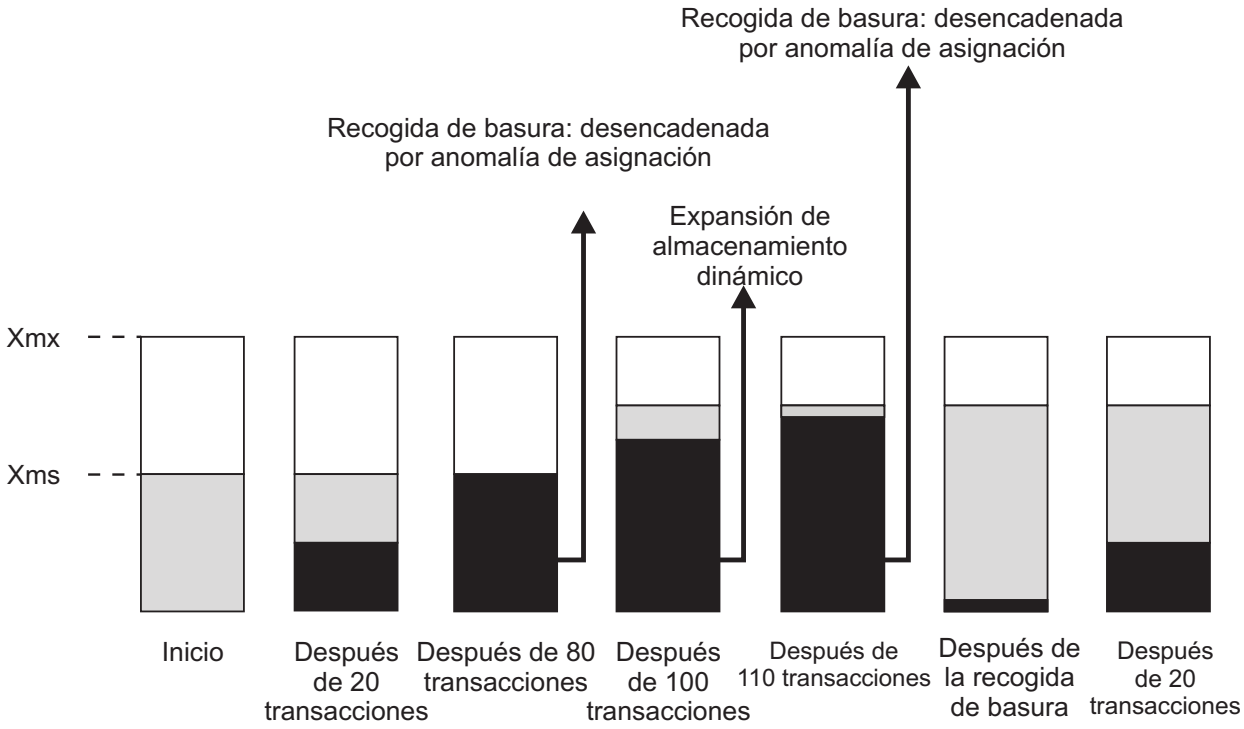


Figura 8. Almacenamiento dinámico en un servidor de JVM con grandes cantidades de basura

Durante las primeras 20 transacciones, la parte activa del almacenamiento dinámico comienza a llenarse. Después de 80 transacciones, el almacenamiento dinámico está lleno y se produce un fallo en la asignación, que desencadena una menor recogida de basura en la JVM. La recogida de basura limpia los objetos con duración corta. Sin embargo, como las solicitudes de la aplicación aún se está ejecutando, se sigue haciendo referencia a parte de los objetos, por lo que no son elegibles para recogida de basura.

Después de 100 transacciones, el colector de basura no puede encontrar suficiente espacio para todos los objetos necesarios actualmente, expande el almacenamiento dinámico. Parte del almacenamiento de la cantidad máxima de almacenamiento reservado para el almacenamiento dinámico (la cantidad especificada por la opción `-Xmx`) se añade a la parte activa del almacenamiento dinámico. La aplicación sigue produciendo objetos, pero la expansión de almacenamiento dinámico ya ha creado suficiente espacio, por lo que la transacción se puede completar.

Después de 110 transacciones, el almacenamiento dinámico ya está ocupado en gran parte. Se produce otro fallo en la asignación que desencadena una mayor recogida de basura. La JVM limpia muchos de los objetos con una duración más

larga utilizados por las transacciones anteriores. Después de otras 20 transacciones, el almacenamiento dinámico comienza a llenarse de nuevo.

Al utilizar la política `gencon` , muchas de las colecciones de recogida de basura secundarias pueden gestionar el tamaño de almacenamiento dinámico antes de que se produzca una recogida de basura principal. Puede descubrir cuántas recogidas de basura se han producido, la ocupación del almacenamiento dinámico y demás información utilizando las estadísticas del servidor de JVM.

Mejora del rendimiento del servidor de JVM

Para mejorar el rendimiento de las aplicaciones que se ejecutan en un servidor de JVM, puede ajustar distintas partes del entorno, incluidas la recogida de basura y el tamaño del almacenamiento dinámico.

Acerca de esta tarea

CICS proporciona informes de estadísticas sobre el servidor de JVM, que incluyen detalles sobre cuánto esperan las tareas a las hebras, tamaños de almacenamiento dinámico, frecuencia de recogida de basura y uso de procesador. También puede utilizar herramientas adicionales de IBM que supervisan y analizan la JVM directamente para ajustar servidores de JVM y ayudar con el diagnóstico de problemas. Puede utilizar las estadísticas para comprobar si la JVM funciona eficazmente, en especial que los tamaños de almacenamiento dinámico son adecuados y la recogida de basura está optimizada.

Procedimiento

1. Compruebe la cantidad de tiempo de procesador que utiliza el servidor de JVM. Las estadísticas del asignador pueden indicarle cuánto tiempo de procesador utilizan los TCB T8. Las estadísticas del servidor de JVM le indican cuánto dedica la JVM a la recogida de basura y cuántas recogidas de basura se producen. Los tiempos de respuesta de la aplicación y el uso de procesador se pueden ver afectados negativamente por la recogida de basura de la JVM.
2. Asegúrese de que tiene suficiente almacenamiento disponible para ajustar los tamaños de almacenamiento dinámico que necesita el servidor de JVM.
3. Ajuste la recogida de basura y el almacenamiento dinámico de la JVM. Un almacenamiento dinámico pequeño puede producir recogidas de basura muy frecuentes, pero uno muy grande causar un uso ineficaz del almacenamiento del MVS. Puede utilizar el IBM Health Center para visualizar y ajustar la recogida de basura y ajustar el almacenamiento dinámico en función de esto.

Qué hacer a continuación

Para obtener un análisis más detallado del uso de la memoria y de los tamaños de almacenamiento dinámico, puede utilizar la herramienta Memory Analyzer en IBM Support Assistant para analizar la memoria de almacenamiento dinámico de Java utilizando instantáneas de volcado del sistema Java o de volcado de almacenamiento dinámico de un proceso de Java.

Examen del uso del procesador por parte de los servidores de JVM

Puede utilizar el recurso de supervisión de CICS para supervisar el tiempo de procesador que utilizan las transacciones que se ejecutan en un servidor de JVM. Todas las hebras de una JVM se ejecutan en bloques de control de tareas (TCB) T8.

Acerca de esta tarea

Puede utilizar el programa de utilidad DFH$MOLS para imprimir registros SMF o utilizar una herramienta como CICS Performance Analyzer para analizar los registros SMF.

Procedimiento

1. Active la supervisión en la región CICS para recopilar la clase de rendimiento de los datos de supervisión.
2. Compruebe el grupo de datos de rendimiento DFHTASK. En concreto puede mirar los siguientes campos:

ID de campo	Nombre del campo	Descripción
283	MAXTTDLY	El tiempo transcurrido durante el que la tarea de usuario esperó para obtener un TCB T8, porque la región CICS había alcanzado el límite de hebras disponibles. El límite de hebras es de 1024 para cada región CICS, y cada servidor de JVM puede tener hasta 256 hebras.
400	T8CPUT	El tiempo de procesador durante el cuál la tarea de usuario estuvo asignada por el dominio del asignador de CICS en un TCB en modo T8 de CICS. Cuando una hebra se asigna a un TCB T8, ese mismo TCB permanece asociado con la hebra hasta que el proceso finaliza.
401	JVMTHDWT	Tiempo transcurrido que la tarea de usuario ha esperado para obtener una hebra de servidor de JVM porque el sistema CICS ha alcanzado el límite de hebras para un servidor de JVM de la región CICS.

3. Para mejorar el uso de procesador, reduzca o elimine el uso de rastreo si es posible.
   a. En un entorno de producción, piense en la posibilidad de ejecutar la región CICS con el distintivo de sistema maestro de CICS desactivado. Tener este distintivo activado aumenta significativamente el coste de procesador de la ejecución de un programa Java. Puede desactivar el distintivo inicializando CICS con SYSTR=OFF, o utilizando la transacción CETR.
   b. Asegúrese de activar el rastreo de JVM únicamente para transacciones especiales. El rastreo de JVM puede producir grandes cantidades de salida en muy poco tiempo y aumentar el coste de procesador. Para obtener más información sobre cómo controlar el rastreo de JVM, consulte “Diagnóstico para Java” en la página 194.
4. No utilice la opción `USEROUTPUTCLASS` en los perfiles de JVM de un entorno de producción. Especificar esta opción tiene un efecto negativo sobre el rendimiento de las JVM. La opción `USEROUTPUTCLASS` permite a los desarrolladores utilizar la misma región CICS para separar la salida de JVM y dirigirla a un destino adecuado, pero implica la creación y la invocación de instancias de clase adicionales.

Cálculo de los requisitos de almacenamiento para servidores de JVM

Para aumentar el número de servidores de JVM en una región CICS, debe asegurarse de que hay suficiente almacenamiento disponible para CICS.

Acerca de esta tarea

Las JVM utilizan almacenamiento por debajo de la línea de los 16 MB, almacenamiento de 31 bits y almacenamiento de 64 bits. Ejecutar un servidor de JVM incurre en un coste de almacenamiento único, no importa cuántas JVM se ejecuten en la región CICS. Cada servidor de JVM y su enclave de Language Environment también requieren una cierta cantidad de almacenamiento de 31 bits y de 64 bits. Los tamaños de almacenamiento dinámico de JVM los gestiona la JVM y CICS utiliza los valores predeterminados. Puede ajustar el tamaño de almacenamiento dinámico si es necesario como parte del ajuste del entorno.

El almacenamiento necesario para el almacenamiento dinámico de la JVM procede del almacenamiento de la región CICS (del almacenamiento de MVS, no del almacenamiento de EDSA). Los almacenamientos dinámicos de JVM mayores reducen el número de JVM que pueden estar presentes en una región CICS y aumentan el tamaño necesario de la región para soportarlas. Sin embargo, si el tamaño de almacenamiento dinámico se define demasiado pequeño, se produce una recogida de basura excesiva, lo que afecta al rendimiento. Puede ajustar las opciones de almacenamiento de la JVM para lograr el mejor rendimiento para las cargas de trabajo de Java. Las opciones de almacenamiento de JVM ayudan a determinar el uso de procesador, el uso de almacenamiento y los tiempos de respuesta de tarea para aplicaciones Java.

Procedimiento

1. Determine la cantidad de almacenamiento libre disponible por debajo del límite utilizando el programa de estadísticas de ejemplo DFH0STAT. Los informes de almacenamiento incluyen la cantidad de almacenamiento de usuario asignado en almacenamiento de 31 bits y por debajo de la línea de los 16 MB.
   - Si no tiene servidores de JVM en ejecución, reste el almacenamiento que se reserva para la región de biblioteca compartida de z/OS de la cantidad total de almacenamiento libre en el espacio de direcciones de CICS. El almacenamiento lo controla el parámetro **SHRLIBRGNSIZE** en MVS y se asigna una vez que se inicia la primera JVM en la región.
   - Si tiene servidores de JVM en ejecución, reste el valor del parámetro **SHRLIBRGNSIZE** de la cantidad total de almacenamiento libre. Cada JVM en ejecución utiliza 12 KB de almacenamiento por debajo de la línea de los 16 MB. El enclave de Language Environment para cada JVM utiliza almacenamiento de 31 bits para el almacenamiento dinámico y el almacenamiento dinámico de biblioteca. La cantidad de almacenamiento de 31 bits asignado se define mediante las opciones `HEAP64` y `LIBHEAP64` de DFHOSGI. También debe restar estos valores de la cantidad total de almacenamiento libre para saber cuánto almacenamiento hay disponible actualmente.

   Si desea cambiar los valores de almacenamiento de 31 bits, puede ajustar el parámetro **SHRLIBRGNSIZE** y las opciones de Language Environment. Consulte "Ajuste de la región de biblioteca compartida de z/OS" en la página 191 y "Utilización de DFHAXRO para modificar el enclave de un servidor de JVM" en la página 186.
2. Calcule cuánto almacenamiento de 64 bits es necesario para cada servidor de JVM adicional. Puede calcular los requisitos de almacenamiento de 64 bits para un servidor de JVM sumando los siguientes requisitos de almacenamiento:
   - El valor de **-Xmx**. El valor predeterminado para este parámetro lo define la JVM, así que debe comprobar la documentación del Information Center de Java.

- El valor del almacenamiento de 64 bits que se asigna mediante la opción HEAP64 en DFHAXRO.
- El valor del almacenamiento de 64 bits que se asigna mediante la opción LIBHEAP64 en DFHAXRO.
- El valor del almacenamiento de 64 bits que se asigna mediante la opción STACK64 en DFHAXRO. Multiplique este valor por el número de hebras permitidas en el servidor de JVM. Para calcular el número de hebras permitidas, añada el valor de atributo THREADLIMIT en el recurso JVMSERVER al valor del parámetro **-Xgcthreads**. Esta opción Java controla el número de hebras del ayudante de recogida de basura de la JVM.

3. Compruebe el valor de **MEMLIMIT** para determinar si tiene suficiente almacenamiento de 64 bits disponible para ejecutar más servidores de JVM. Debe pensar en los demás recursos de CICS que utilizan almacenamiento de 64 bits.

   El parámetro **MEMLIMIT** de z/OS limita la cantidad de almacenamiento de 64 bits (sobre el límite) para la región CICS. Para obtener información sobre los recursos de CICS que utilizan almacenamiento de 64 bits y cómo comprobar y ajustar este parámetro, consulte Estimación, comprobación y configuración de MEMLIMIT en la Guía de rendimiento.

Ajuste del almacenamiento dinámico y de la recogida de basura del servidor de JVM

La recogida de basura de un servidor de JVM lo gestiona la JVM de forma automática. Puede ajustar el proceso de recogida de basura y el tamaño de almacenamiento dinámico para asegurarse de que los tiempos de respuesta y el uso de procesador son óptimos.

Acerca de esta tarea

El proceso de recogida de basura afecta a los tiempos de respuesta de la aplicación y al uso de procesador. La recogida de basura detiene temporalmente todo el trabajo en la JVM, por lo que puede afectar a los tiempos de respuesta de la aplicación. Si se define un tamaño de almacenamiento dinámico pequeño, se puede ahorrar memoria, pero tal vez produzca recogidas de basura más frecuentes y un mayor empleo de tiempo de procesador en dicha recogida. Si se define un tamaño de almacenamiento dinámico demasiado grande, la JVM hace un uso ineficaz del almacenamiento MVS, lo que puede producir anomalías de la memoria caché de datos e incluso transferencia de páginas. CICS proporciona estadísticas que se pueden utilizar para analizar el servidor de JVM. También se puede utilizar el IBM Health Center, que tiene la ventaja de analizar los datos él mismo y recomendar opciones de ajuste.

Procedimiento

1. Recopile estadísticas del servidor y el asignador de JVM durante un intervalo adecuado. Las estadísticas del servidor de JVM pueden indicarle cuántas recogidas de basura importantes y menores se producen y la cantidad de tiempo de procesador que se dedica a dicha recogida. Las estadísticas de asignador pueden informar sobre el uso de procesador para los TCB T8 en la región CICS.
2. Utilice las estadísticas de modalidad TCB del asignador para los TCB T8 para encontrar cuánto tiempo de procesador se dedica a las hebras de servidor de JVM. El campo "Accum CPU Time / TCB" (Tiempo acumulado de CPU/TCB) muestra el tiempo de procesador acumulado necesario para todos los TCB que están conectados a esta modalidad de TCB o lo han estado. El campo "TCB

attaches" (Adjuntos de TCB) muestra el número de TCB T8 que se han utilizado en el intervalo de estadísticas. Utilice estos números para averiguar aproximadamente cuánto tiempo de procesador ha utilizado cada TCB T8.

3. Utilice las estadísticas del servidor de JVM para encontrar el porcentaje de tiempo que se dedica a recogida de basura. Divida el tiempo del intervalo de estadísticas por cuánto tiempo transcurrido se dedica a recogida de basura. Intente lograr un uso de procesador inferior al 2% en recogida de basura. Si el porcentaje es mayor, puede aumentar el tamaño del almacenamiento dinámico para que la recogida de basura se produzca con menos frecuencia.
4. Compare el campo "Current heap size" (Tamaño de almacenamiento dinámico actual) con el campo "GC heap occupancy" (Ocupación de almacenamiento dinámico en recogida de basura) para encontrar cuántos datos activos se están utilizando en el almacenamiento dinámico. Si el almacenamiento dinámico es grande, incluso tras una recogida de basura, el tiempo de detención para realizar dicha recogida es más largo.

   La política optthruput utiliza un único almacenamiento dinámico, lo que puede provocar tiempos de detención menos frecuentes pero más largos. La política gencon parte el almacenamiento dinámico en dos, por lo que la JVM realiza recogida de basura menor en el almacenamiento parcial y mayor en el almacenamiento dinámico completo. La política gencon ayuda a minimizar el tiempo que se dedica a cada detención de recogida de basura.

   Si desea mejorar los tiempos de respuesta de la aplicación, utilice la política gencon:

   a. Consulte las estadísticas del recurso JVMSERVER para encontrar qué política utiliza la JVM.
   b. Si la JVM utiliza optthruput, edite el perfil de JVM para añadir la opción **-Xgcpolicy**. Especifique `-Xgcpolicy:gencon`.
   c. Inhabilite y habilite el recurso JVMSERVER para detectar los cambios en el perfil de JVM. Puede consultar el recurso JVMSERVER habilitado para confirmar que los cambios se han aplicado.
5. Divida el valor liberado del almacenamiento dinámico por el número de transacciones que se han ejecutado en el intervalo para descubrir cuánta basura se ha recogido por cada transacción. Puede averiguar cuántas transacciones se han ejecutado mirando las estadísticas del asignador para TCB T8. Cada hebra de un servidor de JVM utiliza un TCB T8.
6. Opcional: Para realizar un análisis más detallado, añada la opción **-verbose:gc** al perfil de JVM. La JVM graba mensajes de recogida de basura en XML para el archivo especificado en la opción STDERR del perfil de JVM. Consulte verbose:gc logging para obtener ejemplos y explicaciones de los mensajes.

   Consejo: Puede utilizar el archivo de la herramienta Memory Analyzer para realizar análisis más detallados.

Resultados

El resultado del ajuste puede variar en función de la carga de trabajo de Java, del nivel de mantenimiento de CICS y del IBM SDK for z/OS, así como de otros factores. Para obtener información más detallada sobre los valores de almacenamiento y de recogida de basura y sobre posibilidades de ajuste para JVM, consulte Java Guía de diagnósticos.

Ajuste del inicio del servidor de JVM en un sysplex

Si experimenta problemas al iniciar muchos servidores de JVM al mismo tiempo en las regiones CICS de un sysplex, podrá mejorar el rendimiento si ajusta el entorno.

Acerca de esta tarea

Cuando un servidor de JVM se inicia, carga un conjunto de bibliotecas en el directorio `/usr/lpp/cicsts/cicsts42/lib`. Si inicia muchos servidores de JVM al mismo tiempo,cada máquina virtual Java puede tardar un tiempo en cargar las bibliotecas necesarias. Es posible que algunos servidores de JVM excedan el tiempo de espera o que tarden mucho en iniciarse. Para reducir el tiempo de lanzamiento, ajuste el entorno.

Procedimiento

1. Monte zFS en modalidad de solo lectura para mejorar el tiempo de acceso a las bibliotecas desde diferentes servidores de JVM en el sysplex.
2. Monte zFS en otro LPAR del sysplex para proporcionar una copia local de las bibliotecas a una región CICS.
3. Cree una memoria caché de clase compartida para los servidores de JVM para cargar las bibliotecas una vez. Para utilizar una memoria caché de clase compartida, añada la opción **-Xshareclasses** al perfil JVM de cada servidor de JVM. Para obtener más detalles sobre esta opción, consulte el apartado Class data sharing between JVMs.
4. Incremente el valor del tiempo de espera excedido para la infraestructura OSGi. El perfil de JVM DFHOSGI incluye la opción `OSGI_FRAMEWORK_TIMEOUT` que especifica el intervalo de tiempo que CICS espera a que se inicie o se apague el servidor de JVM. Si se sobrepasa ese intervalo, el servidor de JVM no podrá inicializarse o cerrarse correctamente. El valor predeterminado es 60 segundos; por lo tanto, incremente este valor al que resulte conveniente para su entorno.

Gestión de la agrupación de JVM para mejorar el rendimiento

Si ajusta los valores de la agrupación de JVM, tal vez pueda reducir el tiempo de respuesta de las transacciones, asegurándose de que no se malgasta tiempo de procesador cuando se utilizan JVM en agrupación. También puede asegurarse de que cada región CICS que ejecute cargas de trabajo de Java contiene el número óptimo de JVM para el tamaño de la región y que, por lo tanto, aprovecha del mejor modo posible el almacenamiento y el tiempo de procesador.

Acerca de esta tarea

El número de JVM en agrupación que una región CICS puede soportar está regulado principalmente por los siguientes factores:

- La cantidad de tiempo de procesador utilizado por las JVM.
- La cantidad de almacenamiento MVS que necesitan las JVM.
- La cantidad de almacenamiento MVS y el tiempo de procesador que están disponibles para el uso de la región CICS.

Para calcular cuántas JVM en agrupación son necesarias para soportar el nivel necesario de rendimiento de transacción, utilice la siguiente fórmula:

```
ETR x tiempo de respuesta = número de JVM
```

donde:

- ETR (tasa de rendimiento externa) es el nivel deseado de rendimiento de transacción.
- El tiempo de respuesta es el tiempo necesario para ejecutar la transacción en una JVM.

El siguiente procedimiento es un proceso sugerido para ajustar la agrupación de JVM.

Procedimiento

1. Averigüe cuánto tiempo esperan las transacciones para adquirir una JVM. Compruebe el campo de estadísticas "Total Max TCB Pool Limit" (Tiempo de retardo máximo total límite de la agrupación de TCB) en las estadísticas de agrupación del TCB del asignador de CICS. Este campo le muestra cuánto tiempo esperaron las transacciones para adquirir una JVM en los momentos en los que se había alcanzado el límite **MAXJVMTCBS** para la agrupación de JVM. También puede utilizar el campo de datos de supervisión de CICS MAXJTDLY (ID de campo 277), en el grupo de datos de rendimiento DFHTASK, para comprobar cuánto tiempo tuvo que esperar una transacción individual para adquirir una JVM.
   a. Si el tiempo de retardo parece bajo, el límite **MAXJVMTCBS** para la agrupación de JVM no se alcanzará muy a menudo. El campo de estadísticas "Times at Max TCB Pool Limit" (Tiempos con límite máximo de agrupación de TCB) de las estadísticas de agrupación de TCB del asignador de CICS muestra cuántas veces se alcanzó el límite. En esta situación, tal vez sea posible reducir el límite de **MAXJVMTCBS** sin provocar un aumento grave del tiempo de retardo de las transacciones.
   b. Si el tiempo de retardo parece alto, divídalo por el campo de estadísticas "Total Attaches delayed by Max TCB Pool Limit" (Adjuntos totales retardados por límite máximo de agrupación de TCB) en las estadísticas de agrupación de TCB del asignador de CICS para ver cuánto tiempo tuvo que esperar cada transacción. El campo "Average Max TCB Pool Limit delay time" (Tiempo de retardo máximo medio de límite de agrupación de TCB) del resumen de estadísticas de agrupación de TCB contiene esta información. Si la agrupación de JVM está normalmente en su límite **MAXJVMTCBS**, a menudo las transacciones esperan al menos un breve periodo para adquirir una JVM. Aumente el límite **MAXJVMTCBS** solamente si cree que el tiempo de retardo para cada transacción es excesivo.
2. Si ha encontrado que el tiempo de retardo que las transacciones esperan para adquirir una JVM es excesivo, compruebe el nivel de utilización del TCB QR. Las llamadas realizadas por un programa Java para servicios de CICS, como el uso de una clase JCICS para acceder a una cola de datos transitoria, requieren un conmutador para el TCB QR. Cuando el TCB QR alcanza un alto nivel de utilización, añadir más JVM puede no producir un aumento en el rendimiento del sistema CICS. Puede comprobar el nivel de utilización del TCB QR mirando el campo de estadísticas "Accum CPU Time / TCB" (Tiempo de CPU acumulado/TCB) para la modalidad QR en las estadísticas de modalidad de TCB del asignador de CICS.
3. Compruebe la cantidad de tiempo de procesador que utilizan las JVM en agrupación en sus propios TCB J8 y J9. Asegúrese de haber detenido todo uso de procesador innecesario. Para obtener más información, consulte "Examen del uso de procesador por las JVM en agrupación" en la página 174.
4. si desea aumentar el número de JVM en agrupación, compare la cantidad de almacenamiento necesario para el soporte de una sola JVM con la cantidad de espacio de almacenamiento que hay disponible (o que puede hacerse que esté

disponible) para la región CICS y calcule el número máximo de JVM que puede soportar la región CICS. Para obtener más información, consulte el apartado "Cálculo de los requisitos de almacenamiento para JVM en agrupación" en la página 177.

5. Utilice sus hallazgos sobre uso de procesador y disponibilidad de almacenamiento para definir un límite **MAXJVMTCBS** adecuado para la región CICS. Puede cambiar el valor de **MAXJVMTCBS** sin reiniciar CICS utilizando el mandato CEMT SET DISPATCHER.
6. Si recibe un mensaje DFHSJ0203 y un código de retorno de 12 de Language Environment, examine los valores de almacenamiento para las JVM. Las JVM utilizan almacenamiento de 64 bits, y el límite para el almacenamiento de 64 bits para la región CICS lo controla el parámetro **MEMLIMIT** de z/OS. Puede ajustar el valor de almacenamiento, el límite **MAXJVMTCBS** o ambos para reducir la cantidad de almacenamiento que las JVM utilizan en la región CICS. Para obtener más información, consulte "Gestión de las restricciones de almacenamiento de MVS" en la página 181.
7. Si descubre que se produce un número excesivo de no coincidencias y robos en la agrupación de JVM, puede utilizar algunas estrategias para reducirlo. Para obtener más información, consulte "Gestión de no coincidencias y robos" en la página 182.
8. Si la carga de trabajo de Java es regular, predecible e implica un número limitado de distintos perfiles de JVM, puede lanzar las JVM de forma manual antes que las demanden las aplicaciones. Esta estrategia puede reducir el tiempo de retardo para las aplicaciones en periodos en los que aumenta la carga de trabajo. Para obtener más información, consulte Lanzar y terminar JVM e inhabilitar la agrupación JVM de forma manual.

Examen del uso de procesador por las JVM en agrupación

Puede utilizar el recurso de supervisión de CICS para supervisar el tiempo de procesador utilizado por una transacción que invoque un programa de JVM, incluida la cantidad de tiempo de procesador utilizado por un programa de JVM. El recurso de supervisión de CICS también incluye el tiempo transcurrido en la JVM, y el número de solicitudes de API de JCICS emitido por el programa de JVM.

Acerca de esta tarea

Como primera paso para ajustar las JVM, asegúrese de que estas no utilizan tiempo de procesador innecesario. Puede emplear el programa de utilidad DFH$MOLS para imprimir registros SMF o una herramienta como CICS Performance Analyzer para analizar los registros SMF.

Procedimiento

1. Active la supervisión en la región CICS para recopilar la clase performance (rendimiento) de los datos de supervisión.
2. Compruebe los grupos de datos de rendimiento DFHTASK y DFHCICS. En concreto puede mirar los siguientes campos:

Tabla 12. Campos de datos de supervisión relacionados con JVM

Grupo	ID de campo	Nombre del campo	Descripción
DFHTASK	253	JVMTIME	Tiempo transcurrido total utilizado en la JVM por la tarea de usuario. Esto comprende el tiempo de inicialización de la JVM, el tiempo de ejecución de la aplicación Java y el tiempo de limpieza de la JVM. Los campos JVMITIME y JVMRTIME muestran el tiempo de inicialización y de limpieza respectivamente.
DFHTASK	254	JVMSUSP	El tiempo transcurrido durante el que la tarea de usuario fue suspendida por el asignador de CICS mientras se ejecutaba en la JVM.
DFHTASK	260	J8CPUT	El tiempo de procesador durante el que la tarea de usuario fue asignada por el dominio de asignador de CICS a un TCB de modalidad J8 de CICS (utilizado para JVM en la clave de CICS). El campo JVMTIME muestra el tiempo transcurrido real utilizado en la JVM.
DFHTASK	267	J9CPUT	El tiempo de procesador durante el que la tarea de usuario fue asignada por el dominio del asignador de CICS a un TCB de modalidad J9 de CICS (utilizado para JVM en la clave de usuario). El campo JVMTIME muestra el tiempo transcurrido real utilizado en la JVM.
DFHTASK	273	JVMITIME	Tiempo transcurrido utilizado en la inicialización del entorno de JVM. La primera JVM que se inicializa en una región CICS, sea del tipo que sea, tiene un tiempo de inicialización mayor que las JVM posteriores inicializadas en dicha región, debido a la configuración necesaria en ese momento.
DFHTASK	275	JVMRTIME	El tiempo transcurrido dedicado a limpiar la JVM tras su uso por un programa Java. Esto no incluye las recogidas de basura planificadas por CICS, que tienen lugar en una transacción separada (CJGC).
DFHTASK	277	MAXJTDLY	El tiempo transcurrido durante el que la tarea de usuario esperó para obtener un TCB (en modalidad J8 o J9) de JVM de CICS, porque el sistema CICS había alcanzado el límite establecido por el parámetro de inicialización del sistema **MAXJVMTCBS** .
DFHCICS	025	CFCAPICT	El número de solicitudes de la clase base OO de CICS, incluyendo la API de Java para clases de CICS (JCICS), emitidas por la tarea de usuario.

3. Para mejorar el uso de procesador, reduzca o elimine el uso de rastreo si es posible.
   a. En un entorno de producción, ejecutar la región CICS con el distintivo de sistema maestro de CICS desactivado. Tener este distintivo activado aumenta significativamente el coste de procesador de la ejecución de un

programa Java. Puede desactivar el distintivo inicializando CICS con SYSTR=OFF, o utilizando la transacción CETR.

b. Asegúrese de activar únicamente el rastreo de JVM para transacciones especiales. El rastreo de JVM puede producir grandes cantidades de salida en muy poco tiempo y aumentar el coste de procesador. "Diagnóstico para Java" en la página 194 le indica cómo controlar el rastreo de JVM.

4. No utilice la opción `USEROUTPUTCLASS` en los perfiles de JVM de un entorno de producción. Especificar esta opción tiene un efecto negativo sobre el rendimiento de las JVM. La opción `USEROUTPUTCLASS` permite a los desarrolladores utilizar la misma región CICS para separar la salida de JVM y dirigirla a un destino adecuado, pero implica la creación y la invocación de instancias de clase adicionales. Para lograr el mejor rendimiento en un entorno de producción, no utilice esta opción; resérvela para su uso durante el desarrollo de aplicaciones. Los perfiles de JVM proporcionados por CICS no especifican la opción `USEROUTPUTCLASS`.
5. Mire los distintos tipos de JVM en agrupación de la región CICS. En concreto, compruebe si tiene alguna JVM de un solo uso. Las JVM de un solo uso utilizan una gran cantidad de tiempo de procesador en comparación con las JVM continuas. Si la aplicación es de enhebramiento seguro, se puede ejecutar en un servidor de JVM. En caso contrario, mueva la aplicación para que se ejecute en una JVM continua en agrupación.

Cómo afectan distintas JVM en agrupación al uso de procesador

Las JVM en agrupación pueden ser continuas o de un solo uso, y las JVM continuas pueden usar, opcionalmente, la memoria caché de clase compartida. Su elección de JVM en agrupación puede tener una gran repercusión en el uso de procesador.

JVM continuas y JVM de un solo uso

Las JVM de un solo uso tienen un rendimiento pobre, en términos de uso de procesador y de rendimiento de transacción, en comparación con las JVM continuas. Para cada invocación de programa se inicializa una nueva JVM, que se destruye tras cada uso, lo que supone costes muy altos de uso de procesador.

El tiempo necesario para inicializar una JVM de un solo uso es algo inferior al tiempo necesario para una JVM continua que no utilice la memoria caché de clase compartida, aunque es más alto que el tiempo necesario para inicializar una JVM continua que sí emplee la memoria caché de clase compartida. Sin embargo, esta inicialización se produce cada vez que se ejecuta un programa en una JVM de un solo uso, lo que aumenta mucho el tiempo de inicialización acumulado y el tiempo de procesador para cada transacción.

No utilice JVM de un solo uso para ejecutar aplicaciones Java en un entorno de producción. Estas solo son beneficiosas para aplicaciones Java que se diseñaran originalmente para ejecutarse en una JVM de un solo uso y que no se hayan adaptado para ejecutarse en una JVM diseñada para su reutilización. Si ejecuta cualquier programa Java en JVM de un solo uso, su primera acción para mejorar el rendimiento debe ser rediseñar estos programas Java para que se puedan ejecutar en JVM continuas.

Puede que en algunas ocasiones sea necesario reinicializar las JVM, lo que supondrá un coste de inicialización. Algunas de estas situaciones las siguientes:

- Tiene una mezcla de aplicaciones Java en la región CICS que utilizan perfiles de JVM distintos, por lo que se producen no coincidencias y robos.

- Tiene picos y bajadas en la carga de trabajo de Java, y durante los momentos de carga de trabajo baja, algunas JVM agotan su tiempo de espera porque están sin usar durante el tiempo suficiente.

Puede utilizar estrategias para evitar la repercusión de estas situaciones o minimizarla, si se producen con demasiada frecuencia en su región CICS.

JVM que utilizan la memoria caché de clase compartida

Las JVM que utilizan la memoria caché de clase compartida tienen un tiempo de inicialización mucho más breve, ya que utilizan clases precargadas que están disponibles en la memoria caché de clase compartida.

En términos del tiempo de procesador utilizado para cada transacción, algunas aplicaciones tienen un rendimiento algo mejor en una JVM continua que utiliza la memoria caché de clase compartida, y algunas aplicaciones tienen un rendimiento algo mejor en una JVM que no utiliza dicha memoria caché de clase compartida. Si el tiempo de procesador para cada transacción es el factor más importante, incluso más que el tiempo de inicialización, pruebe la aplicación en ambos tipos de JVM. Si las JVM deben reinicializarse de vez en cuando, tenga en cuenta en su evaluación el tiempo de inicialización más bajo para una JVM que utilice la memoria caché de clase compartida.

Cálculo de los requisitos de almacenamiento para JVM en agrupación

Para aumentar el número de JVM en agrupación en una región, debe asegurarse de que hay suficiente almacenamiento disponible para CICS.

Acerca de esta tarea

Las JVM utilizan almacenamiento por debajo de la línea de los 16 MB, almacenamiento de 31 bits y almacenamiento de 64 bits. Ejecutar JVM en agrupación incurre en un coste de almacenamiento único, no importa cuántas JVM se ejecuten en la región CICS. Cada JVM en agrupación y su enclave de Language Environment también requieren una cierta cantidad de almacenamiento de 31 bits y de 64 bits. Los valores predeterminados para los tamaños de almacenamiento dinámico de las JVM son:

Tabla 13. Opciones de perfil de JVM para tamaños de almacenamiento dinámico

Descripción	Opción en el perfil de JVM	Valor del perfil de JVM en agrupación
Almacenamiento dinámico: asignación de almacenamiento inicial	`-Xms`	16 MB
Almacenamiento dinámico: tamaño máximo	`-Xmx`	16 MB

El almacenamiento necesario para el almacenamiento dinámico de la JVM procede del almacenamiento de 64 bits de la región CICS por encima del límite. Los almacenamientos dinámicos de JVM mayores reducen el número de JVM que pueden estar presentes en una región CICS y aumentan el tamaño de **MEMLIMIT** que es necesario para soportarlas. Sin embargo, si el tamaño de almacenamiento

dinámico se define demasiado pequeño, se produce una recogida de basura excesiva, lo que afecta al rendimiento. Las opciones de almacenamiento de JVM deben ajustarse para lograr el mejor rendimiento para las cargas de trabajo de Java. Las opciones de almacenamiento de JVM ayudan a determinar el uso de procesador, el uso de almacenamiento y los tiempos de respuesta de tarea para aplicaciones Java.

Procedimiento

1. Determine la cantidad de almacenamiento libre disponible por debajo del límite utilizando el programa de estadísticas de ejemplo DFH0STAT. Los informes de almacenamiento incluyen la cantidad de almacenamiento de usuario asignado en almacenamiento de 31 bits y por debajo de la línea de los 16 MB.
   - Si no tiene ninguna JVM en ejecución, reste el almacenamiento que se reserva para la región de biblioteca compartida de z/OS de la cantidad total de almacenamiento libre en el espacio de direcciones de CICS. El almacenamiento lo controla el parámetro **SHRLIBRGNSIZE** en MVS y se asigna una vez que se inicia la primera JVM en agrupación de la región.
   - Si tiene JVM en agrupación en ejecución, reste el valor del parámetro **SHRLIBRGNSIZE** de la cantidad total de almacenamiento libre. Cada JVM en ejecución utiliza 12 KB de almacenamiento por debajo de la línea de los 16 MB. El enclave de Language Environment para cada JVM utiliza almacenamiento de 31 bits para el almacenamiento dinámico y el almacenamiento dinámico de biblioteca. La cantidad de almacenamiento de 31 bits asignado se define mediante las opciones `HEAP64` y `LIBHEAP64` de DFHJVMRO. También debe restar estos valores para saber cuánto almacenamiento hay disponible actualmente.

   Si desea cambiar los valores de almacenamiento de 31 bits, puede ajustar el parámetro **SHRLIBRGNSIZE** y las opciones de Language Environment. Consulte "Ajuste de la región de biblioteca compartida de z/OS" en la página 191 y "Utilización de DFHJVMRO para modificar el enclave para JVM en agrupación" en la página 190.
2. Calcule cuánto almacenamiento de 64 bits es necesario para cada JVM en agrupación adicional. Puede calcular los requisitos de almacenamiento de 64 bits para una JVM en agrupación sumando los siguientes requisitos de almacenamiento:
   - El valor -Xmx del perfil de JVM.
   - La cantidad del almacenamiento de 64 bits que especifica la opción `HEAP64` en DFHJVMRO.
   - La cantidad del almacenamiento de 64 bits que especifica la opción `LIBHEAP64` en DFHJVMRO.
   - La cantidad del almacenamiento de 64 bits que especifica la opción `STACK64` en DFHJVMRO. Multiplique este valor por 5 para incluir el sistema y las hebras de aplicación que utiliza cada JVM en agrupación.

   Si las aplicaciones utilizan varios perfiles de JVM que especifican distintos tamaños de almacenamiento dinámico, puede calcular el almacenamiento dinámico máximo medio.
   a. Recopile estadísticas sobre los perfiles de JVM para informar del nivel de actividad correspondiente a cada perfil de JVM y el almacenamiento dinámico máximo.
   b. El campo "Total number of requests for this profile" (Número total de solicitudes para este perfil) indica cuántas veces fue solicitado cada tipo de JVM por una aplicación durante el periodo de muestreo, lo que puede reflejar las proporciones de cada tipo de JVM que generalmente está en la

agrupación de JVM. Multiplique el número total de solicitudes de cada perfil de JVM por el requisito de almacenamiento que ha calculado para ese perfil.

c. Añada los resultados correspondientes a todos los perfiles de JVM y divida esta cifra por el número total de solicitudes de JVM en el periodo de muestreo.

3. Compruebe el valor de **MEMLIMIT** para determinar si tiene suficiente almacenamiento de 64 bits disponible para ejecutar más JVM en agrupación. Debe tener en cuenta los demás recursos de CICS que utilizan almacenamiento de 64 bits.

   El parámetro **MEMLIMIT** de z/OS limita la cantidad de almacenamiento de 64 bits (sobre el límite) para la región CICS. Para obtener información sobre los recursos de CICS que utilizan almacenamiento de 64 bits y cómo comprobar y ajustar este parámetro, consulte Estimación, comprobación y configuración de MEMLIMIT en la Guía de rendimiento.

Ajuste de los almacenamientos dinámicos y de la recogida de basura de JVM en agrupación

Las opciones de recogida de basura que se especifiquen en los perfiles de JVM para JVM en agrupación pueden tener una gran influencia en el rendimiento de sus aplicaciones Java. Puede utilizar la salida del proceso de recogida de basura en la JVM para ajustar estos valores.

Acerca de esta tarea

Además de desencadenarse por anomalías de asignación, la recogida de basura puede lanzarla CICS. CICS lanza la recogida de basura utilizando una llamada a System.gc() cuando la utilización de almacenamiento dinámico en la parte activa del almacenamiento dinámico alcanza un límite especificado. El valor predeterminado es 85%, lo que significa que cuando utiliza el 85%, del almacenamiento en la parte activa del almacenamiento dinámico, CICS planifica una recogida de basura. CICS comprueba la utilización de almacenamiento dinámico tras la ejecución de cada programa Java. Si se ha alcanzado el límite, la transacción de recogida de basura CJGC se planifica para ejecutarse en la JVM inmediatamente después de que finalice la utilización actual de la JVM. Sin embargo, entre estas recogidas de basura podrían seguir sucediendo anomalías de asignación si un programa Java comienza a ejecutarse cuando la utilización del almacenamiento dinámico está por debajo del límite, utiliza todo el almacenamiento restante en la parte activa del almacenamiento dinámico y aún necesita más almacenamiento.

Las recogidas de basura planificadas por CICS se llevan a cabo como una transacción del sistema independiente, CJGC. Las recogidas de basura provocadas por anomalías de asignación, sin embargo, tienen lugar mientras una aplicación se está ejecutando en la JVM. Si se produce la recogida de basura mientras se está ejecutando una aplicación, esta retarda la aplicación y se cuenta en las estadísticas de CICS para la transacción de usuario.

Procedimiento

1. Identifique el perfil de JVM correspondiente a la JVM en agrupación que desea ajustar.
2. Edite el perfil de JVM:
   a. En función de sus objetivos de rendimiento, tal vez desee minimizar los tiempos de respuesta de las tareas definiendo la opción GC_HEAP_THRESHOLD

de forma que sea CICS quien lance la recogida de basura en lugar de que esta se produzca por anomalías de asignación. Si no desea que CICS lance la recogida de basura, puede definir `GC_HEAP_THRESHOLD` en 100. Así todas las recogidas de basura son resultado de anomalías de asignación mientras se ejecutan aplicaciones.

b. Especifique la opción `-verbose:gc`. La JVM envía mensajes de recogida de basura al archivo especificado por la opción `STDERR` en el perfil de JVM (el nombre predeterminado es `dfhjvmerr`). El archivo está en el directorio de z/OS UNIX que se especifica mediante la opción `WORK_DIR` en el perfil de JVM. Si es posible, borre este archivo de cualquier mensaje existente (puede suprimir el archivo y se volverá a crear).

c. Si desea examinar el comportamiento normal de la JVM con sus valores de almacenamiento dinámico actuales, no edite los valores máximo y mínimo de tamaño de almacenamiento dinámico. Si intenta determinar unos valores de almacenamiento dinámico más adecuados para este perfil de JVM, especifique los siguientes valores en dicho perfil de JVM:

```
-Xmx100M
-Xms1M
```

El valor `-Xmx` es grande, para que el almacenamiento dinámico se pueda expandir hasta el tamaño que necesite. El valor `-Xms` es pequeño, por lo que el almacenamiento dinámico comienza con un tamaño menor del necesario y se expande hasta el tamaño mínimo necesario para ejecutar la carga de trabajo de Java.

3. Defina el parámetro de inicialización del sistema **MAXJVMTCBS** como 1. Puede hacerlo mientras CICS esté en ejecución utilizando el mandato **CEMT SET DISPATCHER MAXJVMTCBS**. Con los valores predeterminados de los perfiles de JVM de ejemplo proporcionados por CICS, la salida de todas las JVM de la región CICS se dirige al mismo archivo, por lo que en esta situación contar con una única JVM hace que sea más fácil analizar el comportamiento del colector de basura. Como alternativa, puede modificar la opción STDERR del perfil de JVM para especificar archivos de salida individuales para cada JVM.
4. Utilice el mandato **CEMT INQUIRE JVM** para visualizar el contenido de la agrupación de JVM. Si se visualiza cualquier JVM, depure la agrupación de JVM mediante el mandato **CEMT PERFORM JVMPOOL**. Esto garantiza que se vuelva a crear una JVM con el perfil que se desea ajustar, que contará con la opción `-verbose:gc` y con cualquier nuevo valor de almacenamiento dinámico que haya especificado.
5. Mediante TPNS (Teleprocessing Network Simulator) u otro simulador de red, ejecute un gran número de transacciones que sean representativas de la carga de trabajo habitual o pretendida para una JVM con el perfil que se desea ajustar. Como dato orientativo, cada transacción individual tiene que ejecutarse unas 1.000 veces para asegurar que se invoca la mayor parte de la compilación JIT. Sin embargo, si sabe que es muy poco probable que una transacción se ejecute este número de veces para una JVM concreta, en su lugar ejecute la transacción el número máximo esperado de veces.
6. Localice el archivo que contiene la salida de la recogida de basura para su análisis. La salida está en XML. Consulte verbose:gc logging para obtener ejemplos y explicaciones de la salida, incluida una anomalía de asignación que implica expansión de almacenamiento dinámico y la salida de una recogida de basura desencadenada por una llamada a System.gc().

Consejo: Puede utilizar el archivo de la herramienta Memory Analyzer para realizar análisis más detallados.

La salida de la recogida de basura muestra la siguiente información:

- Recogidas de basura que lanzó CICS al alcanzarse el umbral de utilización de almacenamiento dinámico.
- Recogidas de basura debidas a una anomalía de asignación.
- La cantidad de espacio libre en el almacenamiento dinámico, en bytes y como un porcentaje.
- Se muestra la cantidad de espacio libre antes y después de una recogida de basura.
- El tiempo necesario para cada recogida de basura, en milisegundos.
- Veces que se ha producido expansión del almacenamiento dinámico.
- La cantidad que se expandió el almacenamiento dinámico y el nuevo tamaño del mismo en bytes

El tiempo total necesario para una recogida de basura desencadenada por una llamada a System.gc() se visualiza en la salida dos veces. En una se excluye el tiempo necesario para obtener acceso exclusivo a la máquina virtual (VM) y en la otra se incluye. Puede basar su ajuste en cualquiera de estos tiempos totales, pero asegúrese de utilizar el mismo de forma constante y no sume los dos.

Resultados

El resultado de su ajuste puede variar en función de la carga de trabajo de Java, del nivel de mantenimiento de CICS y del IBM SDK for z/OS, así como de otros factores. Para obtener información más detallada sobre los valores de almacenamiento y de recogida de basura y sobre posibilidades de ajuste para JVM, consulte Java Guía de diagnósticos.

Gestión de las restricciones de almacenamiento de MVS

Si CICS intenta crear demasiadas JVM en agrupación para el almacenamiento de MVS disponible, puede producirse una restricción de almacenamiento de MVS. La JVM en agrupación no se lanza, CICS emite el mensaje DFHSJ0203, y Language Environment tiene una anomalía con un código de retorno de 12.

Acerca de esta tarea

Las JVM utilizan almacenamiento de 64 bits, y la cantidad de almacenamiento de 64 bits (superior al nivel) para la región CICS la limita el parámetro **MEMLIMIT** de z/OS. No se puede alterar el valor de este parámetro mientras CICS esté en ejecución; puede especificar un nuevo valor en el siguiente inicio de la región CICS. Por lo tanto, tal vez desee ajustar otros valores de los perfiles de JVM antes de modificar el parámetro **MEMLIMIT**.

Procedimiento

1. Compruebe que los valores de almacenamiento dinámico de los perfiles de JVM no son demasiado altos, especialmente la opción `-Xmx`, que define el tamaño máximo para el almacenamiento. La opción `-Xmx` para cada perfil de JVM que se utiliza en la región CICS se visualiza cuando se recopilan estadísticas para perfiles de JVM. En “Ajuste de los almacenamientos dinámicos y de la recogida de basura de JVM en agrupación” en la página 179 se explica cómo cambiar estos valores.
2. Compruebe si tiene algún problema con un uso pico alto de un perfil de JVM concreto. Si es así, puede pensar en el uso de la técnica descrita en “Gestión de no coincidencias y robos” en la página 182 para limitar el número de transacciones que solicitan una JVM con ese perfil. Esto implica definir las

transacciones que ejecutan programas de JVM que necesitan ese perfil de JVM, en la misma clase de transacción (TRANCLASS), y colocar un límite para esa clase de transacción.

3. Calcule el número de JVM que se desean ejecutar en la región CICS y ajuste el valor del parámetro **MEMLIMIT**. En "Cálculo de los requisitos de almacenamiento para JVM en agrupación" en la página 177 se explica cómo realizar esta tarea. También tiene que ajustar el límite de MAXJVMTCBS apropiadamente.

Gestión de no coincidencias y robos

CICS asigna JVM en agrupación a aplicaciones e intenta evitar no coincidencias y robos siempre que tenga sentido hacerlo. No obstante, si no hay JVM adecuadas y no hay espacio en la agrupación de JVM, CICS puede satisfacer una petición de aplicación mediante una no coincidencia o un robo. Puede utilizar las estadísticas de CICS para ver si la incidencia de las no coincidencias y los robos en la agrupación de JVM es mayor de lo que le gustaría. Hay algunas técnicas que se pueden utilizar para intervenir si es necesario.

Acerca de esta tarea

Cuando una aplicación solicita una JVM, primero CICS intenta encontrar una JVM adecuada que esté disponible para reutilizarse en la agrupación de JVM. Si no hay disponible ninguna JVM con el perfil de JVM y la clave de ejecución correctas y aún no se ha alcanzado el límite MAXJVMTCBS para la agrupación de JVM, CICS puede crear una nueva JVM para la aplicación.

Si no hay JVM adecuadas y no hay espacio en la agrupación de JVM, CICS destruye una JVM disponible y reinicializa la JVM con el perfil y la clave de ejecución correctos. Este proceso se denomina una *no coincidencia* si la JVM se destruye y se reinicializa pero el TCB se mantiene y se reutiliza, y un *robo* si tanto la JVM como el TCB se destruyen y se sustituyen. Antes de permitir una no coincidencia o un robo, CICS utiliza su mecanismo de selección para decidir si merece la pena.

Procedimiento

1. Evalúe la incidencia de las no coincidencias y los robos en la agrupación de JVM. En las estadísticas de modalidad TCB del asignador de CICS, los campos de estadísticas "TCB Mismatches" (No coincidencias de TCB) y "TCB Steals" (Robos de TCB), para las modalidades de TCB J8 y J9, muestran la incidencia general de no coincidencias y robos en la agrupación de JVM. En las estadísticas de CICS para perfiles de JVM, el campo "Number of times this profile stole a TCB" (Número de veces que este perfil robó un TCB) muestra la incidencia combinada de no coincidencias y robos para cada perfil de JVM.
2. No se puede especificar el número de JVM con cada perfil de JVM que CICS mantiene en la agrupación de JVM. Puede limitar indirectamente el número de JVM con un perfil de JVM concreto limitando el número de transacciones que solicitan una JVM con ese perfil.
   a. Defina las transacciones que ejecutan programas de JVM que necesitan ese perfil de JVM, en la misma clase de transacción (TRANCLASS).
   b. Asigne un valor de MAXACTIVE para la TRANCLASS.

   Este valor limita el número de ejecuciones simultáneas de programas de JVM que necesitan ese perfil de JVM, por lo que limita el número máximo de JVM con ese perfil de JVM que hay en la agrupación de JVM en cualquier momento dado.

3. Como alternativa, puede intentar reducir el número de perfiles de JVM distintos que utiliza su región CICS. Cuanto menor sea el número de tipos de JVM, más posibilidades hay de que una JVM existente coincida con una petición de aplicación, por lo que se reduce el número de no coincidencias y robos:
   a. Compruebe que todos sus perfiles de JVM y sus archivos de propiedades de JVM asociados realmente especifican opciones distintas.
   b. Investigue si puede combinar opciones compatibles en distintos perfiles de JVM para crear un único perfil de JVM. Por ejemplo, si encontrara dos perfiles de JVM muy poco utilizados que contuvieran opciones similares, pero uno especificara un tamaño de almacenamiento dinámico mayor, tal vez pudiera combinar estos perfiles en un único perfil de JVM que especificara el tamaño de almacenamiento dinámico mayor. Aunque algunas aplicaciones tal vez usaran una JVM mayor, la reducción en la incidencia de no coincidencias y robos es más beneficiosa.

Almacenamiento del enclave de Language Environment para JVM

Una JVM se ejecuta como un proceso de z/OS UNIX System Services en un enclave de Language Environment que se crea utilizando el módulo de inicialización previa de Language Environment, CELQPIPI. Puede modificar las opciones de tiempo de ejecución para el enclave con el fin de ajustar el almacenamiento que asigna el MVS.

Las JVM utilizan servicios de Language Environment de MVS en lugar de servicios de Language Environment de CICS. Como resultado, todo el almacenamiento que obtienen las JVM es almacenamiento de MVS, que se obtiene mediante llamadas a servicios de Language Environment de MVS. Este almacenamiento reside en el espacio de direcciones de CICS, pero no se incluye en las áreas de almacenamiento dinámico (DSA) de CICS. Todas las JVM que se ejecutan en CICS utilizan almacenamiento de 64 bits.

El enclave de Language Environment para cada JVM debe contener no solo el almacenamiento dinámico de JVM, sino también una cantidad básica de almacenamiento para cada JVM. El coste del almacenamiento básico representa la cantidad de almacenamiento del enclave de Language Environment que se utiliza para la estructura de la JVM. Cuando se calcula el tamaño total de la JVM, el coste del almacenamiento básico se debe añadir al almacenamiento que se utiliza para el almacenamiento dinámico.

Las opciones de tiempo de ejecución de Language Environment se definen mediante DFHAXRO y DFHJVMRO. DFHAXRO proporciona las opciones para un servidor de JVM, mientras que DFHJVMRO proporciona las opciones para las JVM en agrupación. Los valores predeterminados que proporcionan estos programas para el enclave de JVM se muestran en Tabla 14:

Tabla 14. Opciones de tiempo de ejecución de Language Environment utilizadas por CICS para el enclave de JVM

Opciones de tiempo de ejecución de Language Environment	Valores de servidor de JVM	Valores de JVM en agrupación
Almacenamiento dinámico	`HEAP64(100M,4M,KEEP,4M,512K, KEEP,1K,1K,KEEP)`	`HEAP64(8M,2M,KEEP,512K, 512K,KEEP,1K,1K,KEEP)`
Almacenamiento dinámico de biblioteca	`LIBHEAP64(3M,3M)`	`LIBHEAP64(1M,1M,FREE,16K, 4K,FREE,4K,2K,FREE)`

Tabla 14. Opciones de tiempo de ejecución de Language Environment utilizadas por CICS para el enclave de JVM (continuación)

Opciones de tiempo de ejecución de Language Environment	Valores de servidor de JVM	Valores de JVM en agrupación
Marcos de pila de rutina de biblioteca que pueden residir en cualquier lugar del almacenamiento	`STACK64(1M,1M,32M)`	`STACK64(1M,1M,32M)`
Gestión del almacenamiento dinámico opcional para aplicaciones multihebra	`HEAPPOOLS64(ALIGN)`	N/A
Gestión de almacenamiento dinámico opcional para aplicaciones multihebras	`HEAPPOOLS(ALIGN)`	N/A
Cantidad de almacenamiento reservado para la condición de falta de almacenamiento y el contenido inicial de almacenamiento cuando se asigna y se libera	`STORAGE(NONE,NONE,NONE)`	`STORAGE(NONE,NONE,NONE,0K)`

Para obtener información acerca de las opciones de tiempo de ejecución de Language Environment, consulte *Personalización del entorno del lenguaje z/OS*.

Puede sustituir las opciones de tiempo de ejecución de Language Environment:

Opciones para servidores de JVM
En el caso de servidores de JVM, modifique y vuelva a compilar el programa de ejemplo DFHAXRO, que se describe en "Utilización de DFHAXRO para modificar el enclave de un servidor de JVM" en la página 186. Este programa se define en el recurso JVMSERVER, por lo que se pueden utilizar distintas opciones del enclave de Language Environment para servidores de JVM individuales si es necesario. Los valores predeterminados de almacenamiento de Language Environment que controlan el tamaño inicial del almacenamiento dinámico del enclave de Language Environment y los añadidos incrementales al mismo pueden hacer un uso ineficaz del almacenamiento de MVS. Los valores de almacenamiento que proporciona CICS son más eficaces. También se pueden modificar estos valores para que coincidan mejor con el uso de almacenamiento de las JVM. Asegúrese de que los tamaños de almacenamiento dinámico se definen para evitar varias asignaciones y liberaciones de segmentos.

Opciones para JVM en agrupación
En el caso de JVM en agrupación, utilice el módulo sustituible por el usuario DFHJVMRO, que se describe en "Utilización de DFHJVMRO para modificar el enclave para JVM en agrupación" en la página 190.

Para mejorar la utilización del almacenamiento de MVS, utilice DFHJVMRO para definir la asignación inicial de la cantidad de almacenamiento dinámico del enclave de Language Environment a un valor que se aproxime al almacenamiento que utilicen las aplicaciones Java que se ejecutan en JVM en agrupación, con esto como tamaño de almacenamiento dinámico inicial. Los valores creados mediante DFHJVMRO se aplican a todas las JVM de la región CICS, así que piense

en los distintos tamaños de almacenamiento dinámico y costes de almacenamiento básico que puedan tener JVM con distintos perfiles.

Las cantidades de almacenamiento necesarias para una JVM en un enclave de Language Environment pueden requerir cambios en las salidas de instalación, IEALIMIT o IEFUSI, que se utilizan para limitar los tamaños de **REGION** y **MEMLIMIT**. Un posible enfoque es tener una región propietaria de Java (JOR) a la que se dirijan todas las solicitudes de programas Java. Dicha región ejecuta solo cargas de trabajo Java, minimizando la cantidad de almacenamiento de CICS DSA necesario y permitiendo asignar la máxima cantidad de almacenamiento de MVS a las JVM.

Identificación de las necesidades de almacenamiento de Language Environment para servidores de JVM

Puede identificar un valor adecuado para la asignación inicial del almacenamiento dinámico del enclave de Language Environment en un servidor de JVM generando informes de almacenamiento. Generar informes de almacenamiento aumenta los costes de procesador, por lo que deben ejecutarse a una hora adecuada en un entorno de producción.

Acerca de esta tarea

La opción de tiempo de ejecución `HEAP64` de DFHAXRO controla el tamaño de almacenamiento dinámico del enclave de Language Environment para un servidor de JVM. Esta opción incluye valores para almacenamiento de 64 bits y 31 bits. Puede utilizar su propio programa en lugar de DFHAXRO si lo prefiere. El programa se debe especificar en el recurso JVMSERVER.

Procedimiento

1. Defina las opciones `RPTO(ON)` y `RPTS(ON)` en DFHJVMRO. Estas opciones están en comentarios en el código fuente de DFHAXRO que se incluye. Si se especifican estas opciones, Language Environment informa sobre las opciones de almacenamiento y graba un informe de almacenamiento que muestra el almacenamiento real utilizado.
2. Inhabilite el recurso JVMSERVER. El servidor de JVM se cierra y el enclave de Language Environment se elimina.
3. Habilite el recurso JVMSERVER. CICS utiliza las opciones de tiempo de ejecución de Language Environment en DFHAXRO para crear el enclave para el servidor de JVM. La JVM también se lanza.
4. Ejecute las cargas de trabajo de Java en el servidor de JVM para recopilar datos sobre el almacenamiento que utiliza el enclave de Language Environment.
5. Elimine las opciones `RPTO(ON)` y `RPTS(ON)` de DFHAXRO.
6. Inhabilite el recurso JVMSERVER para generar los informes de almacenamiento. Los informes de almacenamiento incluyen una sugerencia para el almacenamiento dinámico inicial del enclave de Language Environment. La entrada “Suggested initial size” (tamaño inicial sugerido) de las estadísticas de almacenamiento dinámico de usuario de 64 bits contiene el valor sugerido y equivale a la cantidad total de almacenamiento dinámico del enclave de Language Environment que utilizara el servidor de JVM.

Resultados

Los informes de almacenamiento se guardan en un archivo `stderr` en z/OS UNIX. El directorio depende de si se ha redirigido la salida para la JVM en el perfil de JVM. Si no hay redirección, el archivo se guarda en el directorio de trabajo

correspondiente a la JVM. Si no se define ningún valor para WORK_DIR en el perfil, el archivo se guarda en el directorio `/tmp`.

Utilice la información de los informes de almacenamiento para seleccionar un valor adecuado para el almacenamiento dinámico inicial del enclave de Language Environment en DFHAXRO. Language Environment puede realizar añadidos al almacenamiento dinámico, pero no puede eliminar el almacenamiento no deseado que se proporciona en la asignación inicial. Asigne suficiente almacenamiento para asegurarse de que el número de segmentos asignados y liberados es mínimo.

También puede utilizar esta técnica para definir los valores iniciales de tamaño e incremento para las opciones de tiempo de ejecución `LIBHEAP64` t `STACK64`.

Ejemplo

El siguiente ejemplo es un informe de almacenamiento de Language Environment:

```
64bit User HEAP statistics:
      Initial size:                                        50M
      Increment size:                                       4M
      Total heap storage used:                          91977408
      Suggested initial size:                              88M
      Successful Get Heap requests:                       2439
      Successful Free Heap requests:                      1619
      Number of segments allocated:                          1
      Number of segments freed:                              0
31bit User HEAP statistics:
      Initial size:                                     524288
      Increment size:                                   524288
      Total heap storage used (sugg. initial size):    8440784
      Successful Get Heap requests:                       1965
      Successful Free Heap requests:                      1904
      Number of segments allocated:                          2
      Number of segments freed:                              0
```

En base a los valores para el almacenamiento dinámico del enclave de Language Environment que se muestran en el ejemplo, puede definir estos valores para almacenamiento dinámico en DFHAXRO:

```
HEAP64(88M,4M,KEEP,10M,512K,KEEP,1K,1K,KEEP)
```

Utilización de DFHAXRO para modificar el enclave de un servidor de JVM

DFHAXRO es un programa de ejemplo que proporciona un conjunto predeterminado de opciones de tiempo de ejecución para el enclave de Language Environment en el que se ejecuta un servidor de JVM. Por ejemplo, define parámetros de asignación de almacenamiento para el almacenamiento dinámico y la pila de la JVM. Para CICS, los valores de almacenamiento que se ofrecen en DFHAXRO son más apropiados que los valores de almacenamiento predeterminados de Language Environment.

Acerca de esta tarea

Puede actualizar el programa de ejemplo para ajustar el enclave de Language Environment o puede escribir su propio programa basado en el ejemplo. El programa se define en el recurso JVMSERVER y se llama durante la fase de inicialización previa de CELQPIPI del enclave de Language Environment que se crea para un servidor de JVM.

Debe escribir el programa en lenguaje ensamblador y no debe convertirse con el conversor de CICS. Las opciones se especifican como series de caracteres que constan de una longitud de serie de 2 bytes seguidos de la opción de tiempo de ejecución. La longitud máxima para todas las opciones de tiempo de ejecución de Language Environment es de 255 bytes, así que utilice la versión abreviada de cada opción y restrinja los cambio a un total de menos de 200 bytes.

Procedimiento

1. Copie el programa DFHAXRO a una nueva ubicación para editar las opciones de tiempo de ejecución. Si se aplica mantenimiento a la región CICS, tal vez desee reflejar los cambios en el programa. El código fuente de DFHAXRO se encuentra en la biblioteca CICSTS42.CICS.SDFHSAMP.
2. Edite las opciones de tiempo de ejecución utilizando la abreviatura para cada opción. La *Guía de programación del entorno del lenguaje z/OS* cuenta con información completa sobre las opciones de tiempo de ejecución de Language Environment.
   - Mantenga el tamaño de la lista de opciones al mínimo para que el proceso sea rápido y porque CICS añade algunas opciones a esta lista.
   - Utilice la opción HEAP64 para especificar la asignación de almacenamiento dinámico inicial.
   - La opción ALL31, la opción POSIX y la opción XPLINK las fuerza CICS. CICS define la opción ABTERMENC en (ABEND) y la opción TRAP en (ON,NOSPIE).
   - La salida producida por las opciones RPTO y RPTS se graba en la cola de datos transitoria CESE.
   - Cualquier opción que produzca salida lo hace en la terminación de cada JVM. Tenga en cuenta el volumen de salida que se puede producir y que se dirigirá a CESE.
3. Utilice el procedimiento de DFHASMVS para compilar el programa.

Resultados

Cuando se habilita el recurso JVMSERVER, CICS crea el enclave de Language Environment utilizando las opciones de tiempo de ejecución que se especificaran en el programa DFHAXRO. CICS comprueba la longitud de las opciones de tiempo de ejecución antes de pasarlas a Language Environment. Si la longitud es mayor que 255 bytes, CICS no intenta lanzar el servidor de JVM y graba mensajes de error en CSMT. CICS no comprueba los valores que se especifican antes de pasarlos a Language Environment.

Identificación de las necesidades de almacenamiento de Language Environment utilizando estadísticas de JVM

Puede utilizar las estadísticas de CICS para ver cuánto almacenamiento dinámico del enclave de Language Environment van a emplear las JVM. El campo “Peak Language Environment heap storage used” (Almacenamiento dinámico pico de Language Environment utilizado) de las estadísticas del perfil de JVM muestra la cantidad pico (o marca de límite superior) de almacenamiento dinámico del enclave de Language Environment que utilizó una JVM con la clave de ejecución y el perfil especificados. Recopilar esta estadística afecta al rendimiento de las JVM, por lo que el proceso debe realizarse en un momento adecuado en un entorno de producción.

Procedimiento

1. Utilice el mandato **EXEC CICS INQUIRE JVMPROFILE** para identificar cada uno de los perfiles de JVM que se usan en la región CICS. (No hay ningún equivalente de CEMT para este mandato).
2. Especifique la opción LEHEAPSTATS=YES en cada uno de los perfiles de JVM que ha identificado.
3. Depure las JVM utilizando el mandato CEMT SET JVMPOOL PHASEOUT (o el mandato equivalente de EXEC CICS), aproximadamente en el momento de un restablecimiento de estadísticas (antes o inmediatamente después). Esto asegura que las estadísticas recopiladas en el siguiente intervalo de estadísticas sean un reflejo más preciso del uso de almacenamiento para las JVM. También asegura que las JVM se recrearán utilizando la opción LEHEAPSTATS=YES.
4. Ejecute una muestra representativa de las transacciones que utilizan las JVM.
5. Recopile las estadísticas de perfil de JVM utilizando el mandato EXEC CICS COLLECT STATISTICS JVMPROFILE o CEMT PERFORM STATISTICS JVMPROFILE o mire las estadísticas del perfil de JVM que se hayan recopilado durante el intervalo de estadísticas.
6. Elimine la opción LEHEAPSTATS=YES de los perfiles de JVM o cámbiela a NO (que es el valor predeterminado).
7. Depure las JVM utilizando el mandato CEMT SET JVMPOOL PHASEOUT para asegurarse de que se han recreado con la opción LEHEAPSTATS=NO.
8. Examine el campo “Peak Language Environment heap storage used” (Almacenamiento dinámico pico de Language Environment utilizado) en las estadísticas del perfil de JVM para cada perfil de JVM.

Resultados

Utilice el valor de “Peak Language Environment heap storage used” para definirlo como el tamaño del almacenamiento dinámico inicial en DFHJVMRO. Si la cantidad máxima de almacenamiento utilizado varía entre perfiles de JVM, seleccione un valor adecuado en base al uso relativo de cada perfil de JVM. Trate de seleccionar un valor que se aproxime al almacenamiento utilizado por la mayoría de las JVM. Language Environment puede realizar añadidos al almacenamiento dinámico, pero no puede eliminar el almacenamiento no deseado que se proporciona en la asignación inicial.

Identificación de las necesidades de almacenamiento de Language Environment utilizando DFHJVMRO

Puede identificar un valor adecuado para la asignación inicial de la cantidad de almacenamiento dinámico del enclave de Language Environment definiendo opciones de tiempo de ejecución adicionales en DFHJVMRO. Estas opciones aumentan los costes de procesador, por lo que deben utilizarse en un momento adecuado en un entorno de producción.

Acerca de esta tarea

La opción de tiempo de ejecución HEAP64 de DFHJVMRO controla el tamaño de almacenamiento dinámico del enclave de Language Environment. Esta opción incluye valores para almacenamiento de 64 bits y 31 bits.

DFHJVMRO no puede identificar el perfil de JVM al que se aplica cada informe de almacenamiento, por lo que este procedimiento solo debe utilizarse para un único

perfil de JVM cada vez, asegurándose de utilizar transacciones que solamente necesiten ese perfil de JVM.

Procedimiento

1. Defina las opciones RPTO(ON) y RPTS(ON) en DFHJVMRO. Estas opciones están en comentarios en el código fuente de DFHJVMRO que se incluye. Si se especifican estas opciones, Language Environment informa sobre las opciones de almacenamiento y graba un informe de almacenamiento que muestra el almacenamiento real utilizado.
2. Purgue las JVM de la agrupación de JVM para asegurarse de que se recrean utilizando las opciones RPTO(ON) y RPTS(ON). Puede utilizar la vista **Operations** > **Java** > **JVM Pools** (Operaciones > Java > Agrupaciones de JVM) de CICS Explorer, o puede utilizar el mandato **CEMT SET JVMPOOL PHASEOUT**.
3. Ejecute una muestra representativa de las transacciones que utilizan las JVM en agrupación con el perfil de JVM que desea examinar. El perfil de JVM correspondiente a un programa se indica en el recurso PROGRAM.
4. Elimine las opciones RPTO(ON) y RPTS(ON) de DFHJVMRO.
5. Purgue las JVM. Los informes de almacenamiento se graban cuando finaliza cada JVM. Los informes de almacenamiento incluyen una sugerencia para el almacenamiento dinámico inicial del enclave de Language Environment. La entrada “Total heap storage used (sugg. initial size)”(Tamaño de almacenamiento dinámico total utilizado (tamaño inicial sugerido) contiene el valor sugerido y equivale a la cantidad total de almacenamiento dinámico del enclave de Language Environment que utilizara la JVM.
6. Examine todos los conjuntos de informes de almacenamiento para comprobar cualquier variación en la cantidad de almacenamiento utilizado.

Resultados

Seleccione un valor adecuado para el almacenamiento dinámico inicial del enclave de Language Environment en DFHJVMRO. Trate de seleccionar un valor que se aproxime al almacenamiento utilizado por la mayoría de las JVM. Language Environment puede realizar añadidos al almacenamiento dinámico, pero no puede eliminar el almacenamiento no deseado que se proporciona en la asignación inicial.

También puede utilizar esta técnica para definir los valores iniciales de tamaño e incremento para las opciones de tiempo de ejecución `LIBHEAP64` y `STACK64`.

Ejemplo

El siguiente ejemplo es un informe de almacenamiento de Language Environment:

```
64bit User HEAP statistics:
   Initial size:                                      8M
   Increment size:                                    2M
   Total heap storage used:                    30573536
   Suggested initial size:                          30M
   Successful Get Heap requests:                  24395
   Successful Free Heap requests:                 12416
   Number of segments allocated:                      7
   Number of segments freed:                          0

31bit User HEAP statistics:
   Initial size:                                 524228
   Increment size:                               524228
   Total heap storage used (sugg. initial size):      1099824
   Successful Get Heap requests:                         599
```

```
Successful Free Heap requests:                    567
Number of segments allocated:                       2
Number of segments freed:                           0
```

En base a los valores para el almacenamiento dinámico del enclave de Language Environment que se muestran en el ejemplo, puede definir estos valores para almacenamiento dinámico en DFHJVMRO:

```
HEAP64(30M,2M,KEEP,1099824,512K,KEEP,1K,1K,KEEP)
```

Utilización de DFHJVMRO para modificar el enclave para JVM en agrupación

DFHJVMRO especifica las opciones de tiempo de ejecución que se utilizan para crear el enclave de Language Environment en el que se ejecuta una JVM en agrupación. Define parámetros de asignación de almacenamiento para el almacenamiento dinámico y la pila, así como otras opciones. Para CICS, los valores de almacenamiento que se ofrecen en DFHJVMRO son más apropiados que los valores de almacenamiento predeterminados de Language Environment.

Acerca de esta tarea

DFHJVMRO es un módulo sustituible por el usuario (URM) que se llama durante la fase de inicialización previa de CELQPIPI del enclave de Language Environment para cada JVM en agrupación. Tal vez desee modificar la versión del programa que se proporciona en las siguientes situaciones:

- Utilice las opciones `RPTO` y `RPTS` para obtener informes sobre el conjunto de opciones de almacenamiento y el almacenamiento real utilizado para JVM.
- Defina valores de almacenamiento dinámico para el enclave que sean distintos de los valores proporcionados. El almacenamiento dinámico de Java se asigna independientemente del almacenamiento dinámico del enclave.
- Si lo solicita el equipo de servicio técnico de IBM, defina otras opciones para obtener información de diagnóstico.

Debe escribir el programa en lenguaje ensamblador y no debe convertirse con el conversor de CICS. Las opciones se especifican como series de caracteres que constan de una longitud de serie de 2 bytes seguidos de la opción de tiempo de ejecución. La longitud máxima para todas las opciones de tiempo de ejecución de Language Environment es de 255 bytes, así que utilice la versión abreviada de cada opción y restrinja los cambio a un total de menos de 200 bytes.

Procedimiento

1. Copie el programa DFHJVMRO a una nueva ubicación para editar las opciones de tiempo de ejecución. Si se aplica mantenimiento a la región CICS, tal vez desee reflejar los cambios en el programa. El código fuente de DFHJVMRO se encuentra en la biblioteca CICSTS42.CICS.SDFHSAMP.
2. Edite las opciones de tiempo de ejecución utilizando la abreviatura para cada opción. El código fuente para DFHJVMRO contiene comentarios con ejemplos de cómo definir estas opciones. La *Guía de programación del entorno del lenguaje z/OS* cuenta con información completa sobre las opciones de tiempo de ejecución de Language Environment.
   - Mantenga el tamaño de la lista de opciones al mínimo para que el proceso sea rápido y porque CICS añade algunas opciones a esta lista.
   - Utilice la opción HEAP64 para especificar la asignación de almacenamiento dinámico inicial.

- CICS fuerza la opción XPLINK, por lo que Language Environment fuerza todas las opciones de ALL31. La opción POSIX pasa de forma predeterminada a ON debido a la opción AMODE(64). La opción ABTERMENC es ABEND de forma predeterminada y CICS define la opción TRAP como (ON,NOSPIE).
- La salida producida por las opciones RPTO y RPTS se graba en la cola de datos transitoria CESE.
- Cualquier opción que produzca salida lo hace en la terminación de cada JVM. Tenga en cuenta el volumen de salida que se puede producir y que se dirigirá a CESE.

3. Compruebe que la longitud de las opciones no supera los 200 caracteres. La longitud máxima es de 255 caracteres, pero CICS añade algunas opciones automáticamente.
4. Utilice el procedimiento de DFHASMVS para compilar el programa.

Resultados

Cuando CICS recibe una solicitud para ejecutar un programa Java, crea el enclave de Language Environment para las JVM en agrupación utilizando las opciones de tiempo de ejecución que se especificaran en el programa DFHJVMRO. CICS comprueba la longitud de las opciones de tiempo de ejecución antes de pasarlas a Language Environment. Si la longitud es mayor que 255 bytes, CICS no intenta lanzar la JVM en agrupación y graba mensajes de error en CSMT. CICS no comprueba los valores que se especifiquen antes de pasarlos a Language Environment.

Ajuste de la región de biblioteca compartida de z/OS

La región de biblioteca compartida es una característica de z/OS que permite que los espacios de direcciones compartan archivos de la biblioteca de enlaces dinámicos (DLL). Esta función permite que las regiones CICS compartan las DLL que necesiten para las JVM, en lugar de que cada región las tenga que cargar individualmente. Esto puede reducir enormemente la cantidad de almacenamiento real utilizado por MVS y el tiempo necesario para que las regiones carguen los archivos.

El almacenamiento reservado en la región de la biblioteca compartida se asigna a cada región CICS cuando se inicia la primera JVM en la región. La cantidad de almacenamiento que se asigna se controla con el parámetro **SHRLIBRGNSIZE** de z/OS, que está en el miembro BPXPRMxx de SYS1.PARMLIB. El mínimo es 16 MB y el valor predeterminado en z/OS es de 64 MB. Puede ajustar la cantidad de almacenamiento que se asigna para la región de biblioteca compartida investigando cuánto espacio necesita, teniendo en cuenta que otras aplicaciones además de CICS pueden utilizar la región de biblioteca compartida y ajustando el parámetro **SHRLIBRGNSIZE** en función de esto.

Si quiere reducir la cantidad de almacenamiento que se asigna para la región de biblioteca compartida, primero compruebe que no ha malgastado espacio en dicha región. Cree la carga de trabajo normal en el sistema z/OS y luego emita el mandato **D OMVS,L** para visualizar las estadísticas de la biblioteca. Si hay espacio sin utilizar en la región de biblioteca compartida, puede reducir el valor de **SHRLIBRGNSIZE** para eliminar este espacio. Si CICS es el único usuario de la región de biblioteca compartida, puede reducir **SHRLIBRGNSIZE** al mínimo de 16 MB, ya que las DLL necesarias para las JVM solo utilizan unos 10 MB de la región.

Si descubre que se está utilizando todo el espacio de la región de biblioteca compartida, pero aún desea reducir la asignación de almacenamiento en las regiones CICS, hay tres cursos de acción posibles que puede tener en cuenta:

1. Es posible definir el tamaño de la región de biblioteca compartida a una cantidad menor de almacenamiento que el que necesita para los archivos. Cuando la región de biblioteca compartida está llena, los archivos se cargan en el almacenamiento privado y no se benefician del recurso de compartición. Si opta por este curso de acción, debe asegurarse de utilizar primero las aplicaciones más importantes para asegurarse de que pueden utilizar la región de biblioteca compartida. Este curso de acción es el más adecuado si la mayor parte del espacio de la región de biblioteca compartida lo utilizan aplicaciones que no son críticas.
2. Las DLL que se colocan en la región de biblioteca compartida son aquellas marcadas con el atributo ampliado +l. Puede eliminar este atributo de algunos de los archivos para evitar que vayan a la región de biblioteca compartida y así reducir la cantidad de almacenamiento que necesita para dicha región. Si elige este curso de acción, seleccione los archivos que se comparten con menos frecuencia y además intente no seleccionar archivos que tenga la extensión .so. Los archivos con la extensión .so, si no se colocan en la región de biblioteca compartida, se comparten mediante bibliotecas compartidas de usuario, y este recurso de compartición es menos eficiente que utilizar la región de biblioteca compartida. Este curso de acción es el más adecuado si la mayor parte del espacio de la región de biblioteca compartida lo utilizan archivos sin la extensión .so.
3. Si elimina el atributo ampliado +l de todos los archivos relacionados con la JVM de CICS, las regiones CICS no utilizan nada la región de biblioteca compartida nada y no se asigna almacenamiento para ella en las regiones CICS. Si elige este curso de acción, no se beneficia del recurso de compartición de la región de biblioteca compartida. Este curso de acción es el más adecuado si otras aplicaciones del sistema z/OS necesitan una región de biblioteca compartida grande y no se desea asignar esta cantidad de almacenamiento en las regiones CICS.

Si opta por eliminar el atributo ampliado +l de cualquiera de los archivos, cuando se sustituyen dichos archivos por versiones nuevas (por ejemplo, durante una actualización del software), recuerde comprobar que las versiones nuevas de los archivos no tienen dicho atributo.

Puede encontrar más información sobre bibliotecas compartidas en z/OS UNIX en el sitio web de z/OS UNIX System Services, en la dirección http://www.ibm.com/servers/eserver/zseries/zos/unix/perform/sharelib.html.

Capítulo 8. Resolución de problemas de aplicaciones Java

Si tiene algún problema en una aplicación Java, puede utilizar los recursos de diagnóstico que proporcionan CICS y la JVM para determinar la causa de dicho problema.

Acerca de esta tarea

CICS proporciona estadísticas, mensajes y rastreo para ayudarle a diagnosticar problemas relacionados con Java. Las herramientas de diagnóstico y las interfaces que se incluyen con Java pueden proporcionarle información más detallada sobre lo que sucede en la JVM que la que proporciona CICS, ya que CICS no tiene conocimiento de muchas de las actividades que se producen en una JVM.

Puede utilizar herramientas gratuitas que realizan análisis en tiempo real y fuera de línea de una JVM, como JConsole e IBM Health Center. Para obtener detalles completos, consulte Using diagnostic tools en la *Java Diagnostics Guide*.

Procedimiento

1. Si no puede lanzar un servidor de JVM o una JVM en agrupación, compruebe que la configuración de la instalación de Java sea correcta. Utilice los mensajes de CICS y cualquier error que haya en el archivo `stderr` correspondiente a la JVM para determinar cuál puede ser la causa del problema.
   a. Compruebe si está instalada la versión correcta del Java SDK y que CICS tiene acceso al mismo en z/OS UNIX. CICS tiene soporte para el IBM 64 bits SDK para z/OS, Java Technology Edition versión 6.0.1.
   b. Compruebe que el parámetro de inicialización del sistema **USSHOME** está definido en la región CICS. Este parámetro especifica el inicio de los archivos en z/OS UNIX.
   c. Compruebe que el parámetro de inicialización del sistema **JVMPROFILEDIR** está definido correctamente en la región CICS. Este parámetro especifica la ubicación de los perfiles de JVM en z/OS UNIX.
   d. Compruebe que la región CICS tiene acceso de lectura y ejecución a los directorios de z/OS UNIX que contienen los perfiles de JVM.
   e. Compruebe que la región CICS tiene acceso de grabación al directorio de trabajo de la JVM. Este directorio se especifica en la opción `WORK_DIR` del perfil de JVM.
   f. Compruebe que la opción `JAVA_HOME` de los perfiles de JVM apunta al directorio que contiene el Java SDK.
   g. Compruebe que SDFJAUTH esté en la concatenación STEPLIB del JCL de inicio de CICS.
   h. Si utiliza archivos DLL de WebSphere MQ o de DB2, compruebe que la versión de 64 bits de estos archivos está disponible para CICS.
   i. Si ha modificado DFHAXRO o DFHJVMRO para configurar el enclave de Language Environment, asegúrese de que las opciones de tiempo de ejecución no superan los 200 bytes y de que dichas opciones son válidas. CICS no valida las opciones que se especifican antes de pasarlas a Language Environment. Consulte SYSOUT para ver si hay mensajes de error de Language Environment.

2. Si la configuración es correcta, recopile información de diagnóstico para determinar qué les sucede a la aplicación y a la JVM.
   a. Añada `PRINT_JVM_OPTIONS=YES` al perfil de JVM. Cuando se especifica esta opción, todas las opciones que se pasan a la JVM en el inicio, incluido el contenido de las vías de acceso de clases, se imprimen en SYSPRINT. La información se produce cada vez que se lanza una JVM con esta opción en su perfil.
   b. Consulte los archivos `dfhjvmout` y `dfhjvmerr` para obtener información y mensajes de error de la JVM. Estos archivos están en el directorio especificado por la opción `WORK_DIR` en el perfil de JVM. Los archivos pueden tener nombres distintos si se cambiaron las opciones STDOUT y STDERR en el perfil de JVM.
3. Si la aplicación sufre anomalías o su rendimiento no es bueno, depure la misma utilizando un depurador JPDA.
4. Si recibe errores de falta de memoria, estos se pueden deber a que el almacenamiento de 64 bits sea insuficiente, la aplicación puede tener una fuga de memoria o el tamaño de almacenamiento dinámico puede ser muy pequeño.
   a. Utilice las estadísticas de CICS o una herramienta como IBM Health Center para supervisar la JVM. Si la aplicación tiene una fuga de memoria, la cantidad de datos activos que quedan tras la recogida de basura aumenta gradualmente con el paso del tiempo hasta que el almacenamiento dinámico se agota. Las estadísticas del servidor de JVM informan del tamaño del almacenamiento dinámico tras la última recogida de basura y de los tamaños máximo y pico de dicho almacenamiento dinámico.
   b. Ejecute los informes de almacenamiento para Language Environment con el fin de averiguar si hay suficiente almacenamiento disponible. Consulte "Almacenamiento del enclave de Language Environment para JVM" en la página 183.

Qué hacer a continuación

Si no puede arreglar la causa del problema, póngase en contacto con el soporte de IBM. Asegúrese de proporcionar la información necesaria, como se indica en MustGather para informar de problemas de Java.

Diagnóstico para Java

Muchas de las fuentes habituales de información de diagnóstico de CICS contienen información que es aplicable a las aplicaciones Java. Además de la información que proporciona CICS, hay distintas interfaces específicas para la JVM que se pueden utilizar para la determinación de problemas.

Herramientas de diagnóstico de CICS para Java

CICS cuenta con estadísticas y datos de supervisión que se pueden recopilar al ejecutar aplicaciones Java. Si se producen errores, las transacciones finalizan de forma anómala y se graban mensajes en el registro correspondiente. Consulte Descripción general de mensajes y códigos de CICS en Mensajes y códigos Vol 2 para obtener una lista de las terminaciones anómalas y los mensajes que se aplican al dominio de JVM (SJ). Los mensajes relacionados con Java tienen el formato DFHSJxxxx.

También puede activar el rastreo, para producir información de diagnóstico adicional. Los puntos de rastreo para el dominio de JVM se indican en Puntos de rastreo del dominio JVM en Entradas de rastreo.

Cuando se inicia la primera JVM en una región CICS tras la inicialización, CICS emite el mensaje DFHSJ0207, que muestra la versión de Java que se utiliza.

El Java SDK brinda herramientas de diagnóstico e interfaces que proporcionan información más detallada sobre lo que sucede en la JVM. Los mensajes y la información de diagnóstico de la JVM se graban en el archivo de registro `stderr` correspondiente a la JVM. Si se encuentra un problema de Java, consulte siempre este archivo. Por ejemplo, si CICS emite un mensaje para indicar que la JVM ha terminado de forma anómala, el archivo de registro `stderr` es la fuente primaria de información de diagnóstico. "Control de la ubicación para las salidas de JVM stdout, stderr y de volcado" en la página 197 indica cómo controlar la ubicación de la salida de la JVM y cómo redirigir mensajes internos de la JVM y salida de aplicaciones Java que se ejecuten en una JVM.

Al desarrollar aplicaciones Java para CICS, es importante tener en cuenta los requisitos para seguridad de hebra y aislamiento de transacción en CICS. Si una aplicación Java funciona correctamente en su primer uso pero no se comporta adecuadamente en usos posteriores, es probable que el problema se deba a problemas de aislamiento. En este caso, utilice el CICS JVM Application Isolation Utility como parte del trabajo de determinación de problemas para ayudar a identificar el problema.

Archivos de diagnóstico de OSGi

La infraestructura OSGi produce archivos de diagnóstico en zFS que se pueden utilizar para ayudar a resolver problemas con paquetes y servicios OSGi en un servidor de JVM:

Memoria caché de OSGi
La memoria caché de OSGi se encuentra en el directorio *$WORK_DIR*/*applid*/*jvmserver*/`configuration/org.eclipse.osgi` del servidor de JVM. *$WORK_DIR* es el directorio de trabajo del servidor de JVM, *applid* es el APPLID de CICS y *jvmserver* es el nombre del recurso JVMSERVER. La memoria caché de OSGi contiene metadatos de infraestructura y otra información necesaria para ejecutar la infraestructura. La memoria caché se sustituye cuando se lanza el servidor de JVM.

Registros de OSGi
Si se produce un error en la infraestructura OSGi, se crea un registro de OSGi en el directorio *$WORK_DIR*/*applid*/*jvmserver*/`configuration/` del servidor de JVM. La extensión del archivo es `.log`. La infraestructura OSGi continúa grabando en el archivo de registro hasta que alcanza un tamaño de 1.000 KB. Tras esto, la infraestructura OSGi crea otro archivo de registro para escribir más mensajes de error. Puede tener hasta diez archivos de registro en el directorio. Una vez que el décimo archivo de registro esté completo, la infraestructura OSGi sobrescribe el archivo de registro más antiguo.

Herramientas de diagnóstico de JVM

La documentación de CICS brinda información sobre algunas de las herramienta e interfaces de diagnóstico Java:

- En "Activación y gestión del rastreo para servidores de JVM" en la página 202 se describe cómo utilizar el rastreo de componentes que proporciona la transacción CETR para rastrear el ciclo de vida del servidor JVM y las tareas que se ejecutan en ella. Los servidores de JVM no utilizan rastreo auxiliar ni de recurso de rastreo generalizado (GTF). En su lugar, el rastreo se graba en un archivar de zFS que recibe un nombre exclusivo para cada servidor de JVM.
- En "Definición y activación del rastreo para JVM en agrupación" en la página 203 se describe cómo se puede utilizar el recurso de rastreo interno de una JVM en agrupación mediante las interfaces que proporciona CICS. El recurso de rastreo interno puede ofrecer rastreo detallado de puntos de entrada, de salida y de suceso en la JVM. Esta información se presenta en la salida como rastreo de CICS.
- En "Depuración de una aplicación Java" en la página 206 se describe cómo se puede utilizar un depurador remoto para recorrer el código de aplicación para una aplicación Java que se ejecute en una JVM. CICS también brinda un conjunto de puntos de intercepción (o "plugins") en el middleware Java de CICS, que permite insertar programas Java adicionales inmediatamente antes y después de que se ejecute el código Java de la aplicación, con fines de depuración, registro y otros. Para obtener más información, consulte "El mecanismo de plugin de JVM de CICS" en la página 207.

Hay disponibles muchas más herramientas e interfaces de diagnóstico para la JVM. Consulte la Java Guía de diagnósticos para obtener información sobre otros recursos que se pueden utilizar para determinación de problemas en JVM. Los siguientes recursos proporcionan información de diagnóstico útil:

- El recurso de rastreo interno de la JVM se puede utilizar directamente, sin tener que pasar por las interfaces provistas por CICS. La *Guía de diagnóstico* contiene información sobre las propiedades del sistema que se pueden utilizar para controlar el recurso de rastreo interno y para enviar información de rastreo de JVM a varios destinos. Puede utilizar estas propiedades del sistema desde cualquier método o clase de la JVM y para encontrar el valor de cualquier parámetro y tipo de retorno en la llamada al método.
- Si sufre fugas de memoria en la JVM, puede solicitar un volcado de almacenamiento dinámico de la JVM. Un volcado de almacenamiento dinámico genera un volcado de todos los objetos activos (que se siguen usando) que se encuentren en el almacenamiento dinámico de la JVM. También puede analizar las fugas de memoria mediante las herramientas de IBM Health Center y Memory Analyzer, disponibles en el IBM Support Assistant. Para obtener más información acerca de las herramientas de Java, consulte IBM Monitoring and Diagnostics Tools for Java.
- El perfilador HPROF, que se incluye en el IBM 64 bits SDK para z/OS, Java Technology Edition, proporciona información de rendimiento para aplicaciones que se ejecutan en la JVM, para que pueda ver qué partes de un programa utilizan más memoria o tiempo de procesador.
- La JVM brinda interfaces para supervisión, creación de perfiles y RAS (fiabilidad, disponibilidad y capacidad de servicio).

Con todas las interfaces, opciones y propiedades del sistema disponibles para la JVM de IBM que no son específicas para el entorno de CICS, utilice la documentación de la JVM de IBM como fuente de información primaria.

Control de la ubicación para las salidas de JVM stdout, stderr y de volcado

La salida de las aplicaciones Java que se ejecutan en una JVM generalmente se graba en los archivos z/OS UNIX que se indican en las opciones STDOUT y STDERR del perfil de JVM correspondiente a la JVM. Los archivos JAVADUMP se graban en el directorio de trabajo de la JVM en z/OS UNIX, y los TDUMP más detallados de Java se graban en el archivo indicado por la opción JAVA_DUMP_TDUMP_PATTERN. La mayoría de estos nombres de archivo se pueden personalizar en el tiempo de ejecución para identificar de forma exclusiva las JVM que los produjeron. Durante el desarrollo de la aplicación, también puede redirigir la salida de la JVM y los mensajes internos de la JVM utilizando una clase Java.

En la configuración estándar para una JVM de CICS, el archivo indicado por la opción STDOUT del perfil de JVM se utiliza para las solicitudes de System.out, mientras que el archivo indicado por la opción STDERR se utiliza para solicitudes de System.err. Los archivos de salida son archivos z/OS UNIX ubicados en el directorio de trabajo indicado por la opción WORK_DIR en el perfil de JVM.

Puede especificar un nombre de archivo fijo para los archivos `stdout` y `stderr`. Sin embargo, si utiliza un nombre de archivo fijo, la salida de todas las JVM que se crearan con dicho perfil de JVM se añade al mismo archivo, y la salida de las distintas JVM se intercala sin cabeceras de registro. Esto no resulta útil para determinación de problemas.

Una opción mejor es especificar un nombre de archivo variable para los archivos `stdout` y `stderr`. Si se hace esto, se puede conseguir que los archivos sean exclusivos para cada JVM individual durante el tiempo de vida de la región CICS. También puede incluir información identificativa adicional.

- El número exclusivo de JVM diferencia la JVM de cualquier otra JVM de la región CICS. El número de JVM utilizado en CICS es el mismo número que se utiliza para identificar la JVM en el entorno z/OS UNIX, donde se conoce como el ID de proceso (PID) para la JVM. Puede especificar este número como parte del nombre de archivo utilizando el símbolo &JVM_NUM; o la opción **-generate**.
- Puede incluir el ID de aplicación de la región CICS en el nombre de archivo utilizando el símbolo &APPLID; o la opción **-generate**.
- Puede incluir una indicación de fecha y hora en el nombre de archivo utilizando la opción **-generate**.

Otra información identificativa en los nombres de archivo incluye los símbolos &DATE; y &TIME;.

&DATE; se sustituye por la fecha actual con la forma Daammdd.

&TIME; se sustituye por la hora actual con la forma Thhmmss.

La ubicación de la salida de los archivos JAVADUMP de la JVM es el directorio de trabajo de z/OS UNIX indicado por la opción WORK_DIR en el perfil de JVM. Los archivos JAVADUMP se identifican de forma exclusiva mediante una indicación de fecha y hora en sus nombres, y los nombres de estos archivos no se pueden personalizar.

La salida de los TDUMP de la JVM, que contienen salida de volcado más detallada (incluido el espacio de direcciones de la JVM), se graba en un destino de conjunto de datos. El nombre del destino lo especifica la opción

JAVA_DUMP_TDUMP_PATTERN del perfil de JVM. Puede utilizar los símbolos &APPLID;, &DATE; &JVM_NUM;, y &TIME; en este valor para hacer que este nombre sea exclusivo para la JVM individual, como se muestra en los perfiles de JVM que CICS proporciona. Tenga en cuenta que, en este contexto, CICS puede tener que modificar el número de JVM para que sea conforme a los estándares de denominación de conjuntos de datos de MVS.

La JVM graba información en su archivo `stderr` cuando genera un JAVADUMP o un TDUMP. La Java Guía de diagnósticos contiene más información sobre el contenido de los archivos JAVADUMP y TDUMP.

Durante el desarrollo de la aplicación, también puede utilizar la opción USEROUTPUTCLASS de un perfil de JVM para indicar una clase Java que intercepte y redirija la salida desde la JVM y los mensajes internos de la JVM. Puede añadir indicaciones de fecha y hora y cabeceras a los registros de salida e identificar la salida de transacciones individuales que se ejecuten en la JVM. CICS proporciona clases de ejemplo que realizan estas tareas. Especificar esta opción tiene un efecto negativo sobre el rendimiento de las JVM, por lo que no debe utilizarse en un entorno de producción.

Redirección de la salida de JVM stdout y stderr

Durante el desarrollo de la aplicación, los desarrolladores pueden utilizar la opción `USEROUTPUTCLASS` para separar su propia salida de JVM stdout y stderr de una región CICS y dirigirla a un destino identificable que elijan. Puede utilizar una clase Java para redirigir la salida, y puede añadir indicaciones de fecha y hora y cabeceras a los registros de salida. La salida de volcado no se puede interceptar mediante este método.

Especificar la opción `USEROUTPUTCLASS` tiene un efecto negativo sobre el rendimiento de las JVM. Para lograr el mejor rendimiento en un entorno de producción, no utilice esta opción.

La salida grabada en System.out() o System.err(), bien por una aplicación o por un código del sistema, puede redirigirla la clase de redirección de salida. Los archivos z/OS UNIX indicados por las opciones `STDOUT` y `STDERR` del perfil de JVM todavía se utilizan para algunos mensajes emitidos por la JVM, o si la clase indicada por la opción `USEROUTPUTCLASS` no puede grabar datos en el destino que pretende. Por lo tanto, aún debe especificar los nombres de archivo adecuados para ellos.

Para utilizar la opción `USEROUTPUTCLASS`, especifique `USEROUTPUTCLASS=[java class]` en un perfil de JVM, indicando la clase Java de su elección. La clase amplía java.io.OutputStream. Los perfiles de JVM de ejemplo contienen la opción comentada `USEROUTPUTCLASS=com.ibm.cics.samples.SJMergedStream`, que asigna un nombre a la clase de ejemplo proporcionada. Elimine el comentario de esta opción para utilizar la clase com.ibm.cics.samples.SJMergedStream para gestionar la salida de las JVM con ese perfil. CICS también proporciona una clase Java de ejemplo alternativa, com.ibm.cics.samples.SJTaskStream.

El código fuente para las clases suministradas se proporciona como ejemplo, de forma que pueda modificar dichas clases como desee o escribir las suyas propias basándose en los ejemplos.

Para las JMV agrupadas, la clase que utilice debe estar presente en un directorio de una vía de acceso de clases adecuada en el perfil de JVM. La clase de ejemplo proporcionada se incluye automáticamente en una vía de acceso de clases

adecuada y no tiene que especificarla en el perfil de JVM. Si proporciona su propia clase de redirección de salida, añada el directorio a la vía de acceso de clase estándar, utilizando la opción CLASSPATH_SUFFIX, en el perfil de JVM donde haya especificado la opción USEROUTPUTCLASS.

En el caso de los servidores de JVM, no tiene que especificar una vía de acceso de clases. Sin embargo, debe empaquetar la clase de redirección de salida como un paquete OSGi para ejecutar la clase en la infraestructura OSGi. Para obtener más información, consulte el apartado "Escritura de clases Java para redirigir las salidas stdout y stderr de JVM" en la página 139.

Clases de ejemplo com.ibm.cics.samples.SJMergedStream y com.ibm.cics.samples.SJTaskStream proporcionadas por CICS

En el caso de aplicaciones Java que se ejecutan en la hebra de proceso inicial (IPT), que pueden realizar solicitudes de CICS, la salida interceptada procedente de la JVM se puede grabar en una cola de datos transitoria y es posible añadir indicaciones de fecha y hora, identificadores de tarea y de transacción y nombres de programa. Esto permite crear un archivo de registro fusionado que contenga la salida de varias JVM. Puede utilizar este archivo de registro para correlacionar la actividad de JVM con la actividad de CICS. La clase de ejemplo proporcionada por CICS, com.ibm.cics.samples.SJMergedStream, está configurada para crear archivos de registro fusionados como este.

La clase com.ibm.cics.samples.SJMergedStream dirige la salida desde la JVM a las colas de datos transitorias CSJO (para la salida stdout) y CSJE (para la salida stderr y los mensajes internos). Estas colas de datos transitorias se proporcionan en el grupo DFHDCTG y se redirigen a CSSL, pero se pueden redefinir si es necesario.

En concreto, tenga en cuenta que la longitud de los mensajes emitidos por la JVM puede variar, y la longitud de registro máxima para la cola CSSL (133 bytes) tal vez no sea suficiente para contener algunos de los mensajes que reciba. Si esto sucede, la clase de redirección de salida emite un mensaje de error y el texto del mensaje puede verse afectado.

Si descubre que recibe mensajes de la JVM más largos de 133 bytes, debe redefinir CSJO y CSJE como colas de datos transitorias distintas. Conviértalas en destinos de partición extra y aumente la longitud de registro para la cola. Puede asignar la cola a un conjunto de datos físicos o a un conjunto de datos de salida. Tal vez encuentre que un conjunto de datos de salida del sistema resulta más adecuado en este caso, ya que entonces no es necesario cerrar la cola para ver la salida. En Recursos TDQUEUE en la Guía de definición de recursos se explica cómo definir colas de datos transitorias. Si se redefinen CSJO y CSJE, asegúrese de que se instalan tan pronto como sea posible durante un inicio en frío, del mismo modo que en el caso de las colas de datos transitorias que se definen en el grupo DFHDCTG.

Si no se puede acceder a las colas de datos transitorias CSJO y CSJE, la salida se graba en los archivos z/OS UNIX */work_dir/applid*/stdout/CSJO y/*work_dir*/*applid*/stderr/CSJE, donde *work_dir* es el directorio especificado en la opción WORK_DIR del perfil de JVM, e *id_apl* es el identificador APPLID asociado con la región CICS. Si estos archivos no están disponibles, la salida se graba en los archivos z/OS UNIX indicados por las opciones STDOUT y STDERR del perfil de JVM.

Además de redirigir la salida, la clase añade una cabecera a cada registro, que contiene la fecha, la hora, APPLID, TRANSID, el número de tarea y el nombre de programa. El resultado son dos archivos de registro fusionados para la salida de la JVM y para los mensajes de error, en los que el origen de la salida y de los mensaje se puede identificar fácilmente.

En el caso de aplicaciones Java que se ejecutan en hebras distintas a la hebra de proceso inicial (IPT), que no pueden realizar solicitudes de CICS, la salida procedente de la JVM no se puede redirigir utilizando recursos de CICS. La clase com.ibm.cics.samples.SJMergedStream sigue interceptando la salida y añade una cabecera a cada registro. A continuación, la salida se graba en los archivos z/OS UNIX */work_dir/applid*/stdout/CSJO y */work_dir/applid*/stderr/CSJE como se describe anteriormente; o, si estos archivos no están disponibles, en los archivos z/OS UNIX indicados por las opciones STDOUT y STDERR del perfil de JVM.

Como alternativa a crear archivos de registro para la salida de su JVM, puede dirigir la salida desde una única tarea a archivos z/OS UNIX y añadir indicaciones de fecha y hora y cabeceras, para proporcionar secuencias de salida que sean específicas para una única tarea. La clase de ejemplo proporcionada por CICS, com.ibm.cics.samples.SJTaskStream, está configurada para hacer esto. La clase redirige la salida correspondiente a cada tarea a dos archivos z/OS UNIX , uno para la salida de stdout y uno para la salida de stderr, que reciben un nombre exclusivo utilizando un número de tarea (con el formato YYYYMMDD.task.*tasknumber*). Los archivos z/OS UNIX se almacenan en el directorio stdout de la salida stdout, o en el directorio stderr de la salida stderr. El proceso es el mismo tanto para las aplicaciones Java que se ejecutan en la IPT como para las aplicaciones Java que se ejecutan en otras hebras.

Cuando las clases de redirección de salida de ejemplo proporcionada encuentran un error, se emiten uno o más mensajes de error que informan de ello. Si el error se produjo mientras se procesaba un mensaje de salida, los mensajes de error se dirigen a System.err y, por lo tanto, son elegibles para redirección. Sin embargo, si el error se produjo mientras se procesaba un mensaje de error, los nuevos mensajes de error se envían al archivo indicado por la opción STDERR en el perfil de JVM. Esto evita un bucle recursivo en la clase Java. Las clases no devuelven excepciones para el programa Java de llamada.

Las clases se envían en el archivo com.ibm.cics.samples.jar, que se encuentra en el directorio /usr/lpp/cicsts/cicsts42/lib, donde /usr/lpp/cicsts/cicsts42 es el directorio de instalación de los archivos de CICS en z/OS UNIX. El código fuente para las clases también se proporciona como ejemplos, de forma que pueda modificar dichas clases como desee o escribir las suyas propias basándose en los ejemplos. Para obtener más información, consulte el apartado “Escritura de clases Java para redirigir las salidas stdout y stderr de JVM” en la página 139.

Control de las opciones de volcado de Java

La opción JAVA_DUMP_OPTS de los perfiles de JVM especifica las opciones de volcado para la JVM.

Puede utilizar esta opción para definir sus opciones de volcado preferidas de Java.

Puede encontrar información acerca de las opciones de volcado de Java en la Java Guía de diagnósticos.

Gestión de los archivos de registro de OSGi de los servidores de JVM

La infraestructura OSGi graba errores en un conjunto de archivos de trabajo del directorio de trabajo del servidor de JVM. Puede gestionar el número y el tamaño de los archivos de registro para cada servidor de JVM y si los valores predeterminados no resultan adecuados para su entorno.

Acerca de esta tarea

La infraestructura OSGi graba errores en un archivo de registro del directorio *$WORK_DIR*/*applid*/*jvmserver*/configuration en zFS, donde *$WORK_DIR* es el directorio de trabajo del servidor de JVM, *applid* es el APPLID de CICS y *jvmserver* es el nombre del recurso JVMSERVER. La infraestructura OSGi continúa grabando en el archivo de registro hasta que alcanza un tamaño de 1.000 KB. Tras esto, la infraestructura OSGi crea otro archivo de registro para escribir más mensajes de error. Puede tener hasta diez archivos de registro en el directorio. Una vez que el décimo archivo de registro esté completo, la infraestructura OSGi graba sobre el archivo de registro más antiguo.Cada servidor de JVM puede tener, por lo tanto, hasta 10.000 KB de almacenamiento asignado a los archivos de registro en zFS.

Puede añadir opciones al perfil de JVM para cambiar el número y el tamaño de los archivos de trabajo utilizados por la infraestructura OSGi para reducir o aumentar el número de archivos y la utilización del almacenamiento.

Procedimiento

- Para cambiar el número máximo de archivos de registro, añada el parámetro **eclipse.log.backup.max** al perfil de JVM.
- Para cambiar el tamaño máximo de cada archivo de registro, añada el parámetro**eclipse.log.size.max** al perfil de JVM.

Ejemplo

El siguiente ejemplo muestra un perfil de JVM con los dos parámetros especificados. En este ejemplo, la infraestructura OSGi puede utilizar hasta cinco archivos de registro y cada archivo de registro tiene un tamaño máximo de 500 KB.

```
#Parameters to control the number and size of OSGi logs
#
eclipse.log.backup.max=5
eclipse.log.size.max=500
#
#
```

Rastreo de componentes de CICS para JVM

Además del rastreo que produce Java, CICS proporciona algunos puntos de rastreo estándar en los dominios SJ (JVM) y AP para los niveles de rastreo 0, 1, y 2. Estos puntos de rastreo rastrean las acciones que CICS realiza al configurar y gestionar servidores de JVM y JVM en agrupación.

Puede activar los puntos de rastreo de los dominios SJ y AP en los niveles 0, 1 y 2 utilizando la transacción CETR. Para obtener detalles de todos los puntos de rastreo estándar en el dominio SJ, consulte Puntos de rastreo del dominio JVM en Entradas de rastreo.

Rastreo de componentes SJ y AP para servidores de JVM

El componente SJ para servidores de JVM rastrea el lanzamiento y el apagado de servidores de JVM. El rastreo de las operaciones de ciclo de vida del servidor de JVM se realiza en la tabla de rastreo interno. El rastreo de las operaciones de ciclo de vida del lanzador de JVM, JVM, y la infraestructura OSGi se realiza en un archivo de zFS. Además, el componente AP rastrea las transacciones que se ejecutan en el servidor de JVM en el mismo archivo de rastreo. Por ejemplo, los sucesos de la infraestructura OSGi se graban en el archivo de rastreo del siguiente modo:

- En el nivel de rastreo 0, la infraestructura OSGi graba errores en el archivo de rastreo.
- En el nivel de rastreo 1, la infraestructura OSGi graba información, avisos y errores en el archivo de rastreo.
- En el nivel de rastreo 2, la infraestructura OSGi graba depuración, información, avisos y errores en el archivo de rastreo.

Si activa el rastreo de componentes AP, se rastreará la siguiente solicitud que entre en la infraestructura OSGi.

Rastreo del componente SJ para JVM en agrupación

El rastreo del dominio SJ para JVM en agrupación rastrea las acciones de CICS asociadas con el lanzamiento y la gestión de JVM en agrupación:

- En un nivel de rastreo 0, CICS rastrea sucesos extraordinarios y errores. A diferencia del rastreo de excepciones de CICS, que no se puede desactivar, el rastreo de JVM de nivel 0 generalmente está desactivado.
- En el nivel de rastreo 1 y 2, se pueden obtener niveles más profundos de rastreo de JVM. Los niveles de punto de rastreo de JVM llegan al 9 y ofrecen detalles en profundidad de los componentes. Al activar el rastreo interno de JVM en un nivel, también se habilita el rastreo para todos los niveles por encima de este. Por ejemplo, si se activa el rastreo del nivel 1 para una transacción, también se recibe rastreo de nivel 0 para esa transacción.

Además, se pueden utilizar niveles de rastreo adicionales para controlar el recurso de rastreo interno de la JVM en agrupación. Para seleccionar un nivel superior al 2, cambie el parámetro de inicialización del sistema **JVMxxxxTRACE**. Por ejemplo, puede especificar el rastreo de nivel 5 como **JVMLEVEL5TRACE**. Si desea crear especificaciones más complejas para el rastreo de JVM que utilicen varios puntos de rastreo, o si no quiere utilizar niveles de punto de rastreo en toda su especificación, utilice el parámetro **JVMUSERTRACE** para crear su propia serie de opciones de rastreo.

Activación y gestión del rastreo para servidores de JVM

Puede activar el rastreo de servidores de JVM activando el rastreo de componentes SJ y AP. En la tabla de rastreos internos se graban pequeñas cantidades de rastreo, pero la mayor parte se graba en un archivo exclusivo en zFS para cada servidor de JVM. Este archivo no se recorta, por lo que su tamaño se debe gestionar en zFS.

Acerca de esta tarea

El servidor de JVM no utiliza rastreo auxiliar ni de recurso de rastreo generalizado (GTF). En su lugar, el rastreo se graba en un archivo de zFS que recibe un nombre exclusivo para cada servidor de JVM. El nombre de archivo predeterminado tiene

el formato *id_appl.servidor_jvm.*dfhjvmtrc y CICS lo crea en el directorio de trabajo para la JVM cuando se habilita el recurso JVMSERVER. Puede cambiar el nombre y la ubicación del archivo de rastreo en el perfil JVM. Si se suprime o se cambia el nombre del archivo de rastreo cuando el servidor de JVM se está ejecutando, CICS no vuelve a crear el archivo y el rastreo no se graba en otro destino.

Procedimiento

1. Utilice la transacción CETR para activar el rastreo para el servidor de JVM. Puede utilizar dos componentes para producir rastreo para un servidor de JVM:
   - Seleccione el componente SJ para rastrear las acciones que realiza CICS para lanzar y detener el servidor de JVM. CICS graba en el archivo de rastreo de zFS durante el lanzamiento del servidor de JVM y en la tabla de rastreos internos durante la conclusión del servidor de JVM.
   - Seleccione el componente AP para rastrear las transacciones que se ejecutan dentro del servidor de JVM. Si se selecciona esta opción, CICS graba en el archivo de rastreo de zFS.
2. Defina el nivel de rastreo para los componentes SJ y AP:
   - SJ y AP de nivel 0 producen rastreo únicamente para excepciones, como errores durante la inicialización del servidor de JVM o problemas en la infraestructura OSGi.
   - SJ y AP de nivel 1 producen información adicional de rastreo, como mensajes de aviso e informativos en la infraestructura OSGi.
   - SJ y AP de nivel 2 producen información de rastreo de depuración, que brinda información más detallada sobre el proceso de servidor de JVM.

   CICS graba el rastreo en el archivo de rastreo de zFS.
3. Vea los resultados de rastreo en el archivo *applid.jvmserver.*dfhjvmtrc. Cada entrada de rastreo tiene una indicación de fecha y hora. Puede cambiar el nombre y la ubicación del archivo de rastreo mediante la opción de perfil JVMTRACE.
4. Para gestionar el tamaño del archivo, puede suprimir las entradas antiguas. Si inhabilita el recurso JVMSERVER, puede suprimir o renombrar el archivo si desea conservar la información por separado. Si habilita el recurso JVMSERVER, CICS añade entradas de rastreo al archivo de rastreo si es que ya existe y crea un archivo en zFS si es que no existe tal archivo de rastreo.

Definición y activación del rastreo para JVM en agrupación

Las máquinas virtuales Java (JVM) producen sus propios puntos de rastreo. Puede controlar el recurso de rastreo interno de la JVM en agrupación mediante interfaces que proporciona CICS. Los puntos de rastreo para las JVM en agrupación de un entorno de CICS generan salida como rastreo de CICS.

Acerca de esta tarea

El dominio SJ utiliza los niveles de rastreo 29 a 32 para controlar el recurso de rastreo interno de la JVM. Estos niveles se corresponden con las opciones de CICS del rastreo de nivel 0 de JVM en agrupación, rastreo de nivel 1 de JVM en agrupación, rastreo de nivel 2 de JVM en agrupación y rastreo de usuario de JVM en agrupación.

Las opciones de rastreo de JVM predeterminadas que se proporcionan en CICS se correlacionan con los niveles de punto de rastreo de nivel 0, nivel 1 y nivel 2 para JVM. La opción de rastreo de usuario de JVM se puede utilizar para especificar niveles de rastreo u opciones de rastreo complejas.

Las opciones de rastreo de JVM se definen utilizando un campo de 240 caracteres "de formato libre". Puede especificar algunos o todos los parámetros siguientes:

- Un nivel de rastreo.
- Un componente, que es un subcomponente de JVM (un área funcional, como un dominio de CICS).
- Un tipo de punto de rastreo o un grupo.
- Un identificador de punto de rastreo.

Puede añadir más parámetros a las especificaciones de CICS para el rastreo de nivel 0 de JVM, rastreo de nivel 1 de JVM y rastreo de nivel 2 de JVM si desea incluir o excluir elementos particulares en los niveles de rastreo seleccionados. Si desea especificar niveles de rastreo más profundos u opciones de rastreo más complejas, utilice la opción de rastreo de usuario de JVM para crear una serie de opciones de rastreo que incluya los parámetro que elija. Tenga en cuenta que las especificaciones de nivel de punto de rastreo no se aplican a puntos de rastreo que se especifiquen explícitamente mediante su ID de punto de rastreo. Esto significa que puede añadir ID de punto de rastreo que estén clasificados en cualquier nivel a cualquiera de las opciones de rastreo de JVM en CICS, y estos se proporcionan cuando se activa la opción, independientemente de los niveles de punto de rastreo que habilite esa opción de rastreo.

La información sobre rastreo de aplicaciones Java y de la JVM en el Java Guía de diagnósticos indica los niveles de rastreo, los componentes, los tipos de punto de rastreo y los grupos de punto de rastreo posibles. Estos parámetros de rastreo dependen de la versión del IBM 64 bits SDK para z/OS, Java Technology Edition que se utilice y también pueden cambiar durante el tiempo de vida de una versión, por lo que debe consultar la versión adecuada de la *Guía de diagnóstico* para conocer la información más reciente.

El archivo de formato de rastreo que se proporciona con el IBM 64 bits SDK para z/OS, Java Technology Edition lista cada punto de rastreo de JVM con su identificador. Para la versión 6.0.1, el archivo se llama `J9TraceFormat.dat`. Puede utilizar este archivo para identificar un punto de rastreo de JVM individual. Este archivo está sujeto a cambios sin previo aviso; se incluye un número de versión como la primera línea del archivo, que se actualiza si se cambia el archivo. Puede encontrar este archivo en el directorio de instalación de Java.

El rastreo de JVM puede producir una gran cantidad de salida, por lo que normalmente solo debe activarlo para transacciones especiales, en lugar de activarlo de forma global para todas las transacciones. Cuando se activan las opciones de rastreo para una transacción, CICS pasa las opciones de rastreo a la JVM en el punto en el que la transacción comienza a utilizar la JVM. El punto de rastreo SJ 052E del nivel 2 del dominio SJ de CICS muestra la serie de opciones que se ha pasado a la JVM. Las opciones de rastreo se aplican únicamente durante la duración de la utilización de la transacción de la JVM.

Procedimiento

- Para definir opciones de rastreo de JVM predeterminadas para todas las JVM de la región CICS puede utilizar los parámetros de inicialización del sistema CICS **JVMLEVEL0TRACE**, **JVMLEVEL1TRACE**, **JVMLEVEL2TRACE** y **JVMUSERTRACE**. Estos

parámetro solo se pueden proporcionar en el inicio de CICS, no es posible definirlos en la macro DFHSIT. Puede utilizar la transacción CETR para visualizar y modificar estas opciones. Estos parámetros no activan el rastreo de JVM, únicamente definen las opciones de rastreo de JVM predeterminadas.

- Para definir o cambiar opciones de rastreo mientras CICS está en ejecución, utilice cualquiera de los siguientes métodos:
  1. Utilice las pantallas de opciones de rastreo de JVM de la transacción CETR. Puede especificar series de opciones de rastreo y especificar si cada nivel de rastreo se aplica para rastreo estándar, rastreo especial o ambos. Para obtener más información, consulte el apartado CETR - control de rastreo en Transacciones suministradas por CICS.
  2. Utilice los mandatos **EXEC CICS INQUIRE JVMPOOL** y **EXEC CICS SET JVMPOOL**. El mandato **INQUIRE JVMPOOL** visualiza las opciones de rastreo de JVM que ha definido para la agrupación de JVM y el mandato **SET JVMPOOL** las cambia. Las opciones de rastreo de JVM no están disponibles en los equivalentes CEMT para estos mandatos.
- Para activar el rastreo de JVM, utilice cualquiera de los siguientes métodos. Recuerde activar el rastreo de JVM únicamente para transacciones especiales.
  1. Utilice la pantalla Transaction and Terminal Trace (Rastreo de transacción y terminal) en la transacción CETR para activar el rastreo especial (o, si es necesario, el rastreo estándar) para las transacciones relevantes. Para obtener más información, consulte el apartado CETR - control de rastreo en Transacciones suministradas por CICS.
  2. Utilice el parámetro de inicialización del sistema CICS **SPCTRSJ** o **STNTRSJ** para activar el rastreo de JVM en el inicio. **SPCTRSJ** se aplica al rastreo especial y **STNTRSJ** se aplica al rastreo estándar. Utilice al parámetro de inicialización del sistema **SPCTRSJ** en lugar del parámetro de inicialización del sistema **STNTRSJ**. Especifique números de nivel del 29 al 32 para activar los niveles de rastreo de JVM que sean necesarios. Estos parámetro solo se pueden proporcionar en el momento de inicio de CICS, no es posible definirlos en la macro DFHSIT.
  3. Utilice el mandato **EXEC CICS SET TRACETYPE** para definir los niveles de rastreo del 29 al 32 para el componente SJ. Utilice la opción `SPECIAL` en lugar de la opción `STANDARD`.

Resultados

Cuando se activa el rastreo de JVM, los resultados aparecen como puntos de rastreo de CICS en el dominio SJ (JVM). Cada punto de rastreo de JVM que se genera aparece como una instancia de un punto de rastreo de CICS:

- SJ 4D02 es el punto de rastreo utilizado para la información de rastreo de JVM con formato.
- SJ 4D01 se utiliza para cualquier punto de rastreo de JVM que CICS no puede formatear. Si ve este punto de rastreo a menudo, compruebe que el archivo de formato de rastreo proporcionado con el IBM 64 bits SDK para z/OS, Java Technology Edition está presente en el subdirectorio `/lib/` de la instalación del SDK. Para la versión 6.0.1, se llama `J9TraceFormat.dat`. CICS necesita este archivo para dar formato a los puntos de rastreo de JVM.

Si el recurso de rastreo de JVM tiene alguna anomalía, CICS emite el punto de rastreo SJ 4D00.

Qué hacer a continuación

En Java Guía de diagnósticos se proporciona información más detallada sobre rastreo de JVM y sobre determinación de problemas para JVM.

Además de las interfaces que proporciona CICS, puede utilizar el recurso de rastreo interno de la JVM directamente. Las propiedades del sistema de JVM son un método válido de definir y activar opciones de rastreo para JVM en un entorno CICS. La *Guía de diagnóstico* contiene más información sobre las propiedades del sistema que se pueden utilizar para controlar el recurso de rastreo interno.

Depuración de una aplicación Java

La JVM en CICS soporta la Java Platform Debugger Architecture (JPDA), que es el mecanismo de depuración estándar incluido en la plataforma Java 2. Esta arquitectura proporciona un conjunto de API que permiten adjuntar un depurador remoto a una JVM.

Acerca de esta tarea

Puede utilizar cualquier herramienta que soporte JDPA para depurar una aplicación Java que se ejecute en CICS. Por ejemplo, puede utilizar el Java Debugger (JDB) que se incluye con el Java SDK en z/OS. Para adjuntar un depurador remoto de JPDA, debe definir algunas opciones en el perfil de la JVM.

No habilite la depuración para perfiles de JVM que especifique CLASSCACHE=YES o el perfil DFHJVMCD.

Procedimiento

1. Añada la opción **-Xdebug** al perfil de JVM para lanzar la JVM en modalidad de depuración. Si el perfil de JVM lo comparten más de una aplicación, puede utilizar un perfil de JVM para depuración.
2. Opcional: Añada la opción **-Xrunjdwp** para especificar los detalles de la conexión entre el depurador y la JVM. Si configura esta opción para depurar un servidor JVM, especifique `suspend=n`. Esta opción hace que CICS no espere a conectar el depurador a la máquina virtual Java antes de finalizar los mandatos o el procesamiento en la región.

   Los depuradores pueden tener requisitos y prestaciones de conexión distintos, por lo que debe consultarse la documentación que proporciona el depurador.
3. Adjunte el depurador a la JVM. Si se produce un error durante la conexión, por ejemplo, un host TCP/IP o un valor de puerto incorrecto, se graban mensajes en las secuencias de salida estándar y de error estándar de la JVM.
4. Mediante el depurador, compruebe el estado inicial de la JVM. Por ejemplo, compruebe la identidad de hebras que se hayan iniciado y de clases de sistema que se hayan cargado. La JVM suspende la ejecución, la aplicación Java aún no se ha iniciado.
5. Defina un punto de interrupción en un punto adecuado de la aplicación Java especificando el nombre de clase completo de Java y el número de línea del código fuente. Como en este punto generalmente no se ha cargado ninguna clase de aplicación, el depurador indica que la activación de este punto de interrupción se aplaza hasta que se cargue la clase. Deje que la JVM se ejecute por el código de middleware de CICS hasta el punto de interrupción de la aplicación, momento en el que se vuelve a suspender la ejecución.

6. Examine las clases cargadas y las variables y defina más puntos de interrupción para avanzar por el código, según sea necesario.
7. Finalice la sesión de depuración. Puede dejar que la aplicación se ejecute hasta su terminación, momento en el que la conexión entre el depurador y la JVM de CICS se cierra. Algunos depuradores soportan la terminación forzada de la JVM, lo que resulta en una terminación anómala (abend) y en mensajes de error en la consola del sistema CICS.

El mecanismo de plugin de JVM de CICS

Además de las interfaces de depuración estándar de JPDA en la JVM, CICS proporciona un conjunto de puntos de intercepción en middleware Java de CICS, que puede ser valioso para desarrolladores que estén depurando aplicaciones. Puede utilizar estos puntos de intercepción (o plugins) para insertar programas Java adicionales inmediatamente antes y después de que se ejecute el código Java de la aplicación.

Los programas de plugin tienen disponible información sobre la aplicación (por ejemplo, nombre de clase y nombre de método). Los programas de plugin también pueden utilizar la API de JCICS para obtener información sobre la aplicación. Estos puntos de intercepción se pueden utilizar junto con las interfaces estándar de JPDA para proporcionar recursos de depuración adicionales específicos de CICS. También se pueden utilizar para fines distintos a depuración, de modo parecido a los puntos de salida de usuario en CICS.

Existen tres tipos de puntos de salida de Java:

- Un plugin de contenedor EJB de CICS que proporciona métodos a los que se llama inmediatamente antes y después de que se invoque un método EJB.
- Un plugin de CORBA de CICS que proporciona métodos a los que se llama antes y después de que se invoque un método CORBA.
- Un plugin de derivador Java de CICS que proporciona métodos a los que se llama inmediatamente antes y después de que se invoque un programa Java.

Puede utilizar los plugins de depuración con JMV en agrupación. Cuando se utilizan programas de plugin para depurar aplicaciones Java, es necesario especificar las clases en la vía de acceso de clase estándar para la JVM que utilizará la aplicación. La vía de acceso de clase estándar se especifica mediante la opción `CLASSPATH_SUFFIX` en el perfil de JVM. Para obtener más información, consulte el apartado "Clases y vías de acceso de clases en las JVM" en la página 9. Añade clases para programas de plugin el mismo modo que las clases para aplicaciones ordinarias.

La interfaz de programación consta de dos interfaces Java.

- **DebugControl** (nombre completo: `com.ibm.cics.server.debug.DebugControl`) define las llamadas a método que se pueden realizar a una implementación proporcionada por el usuario.
- **Plugin** (nombre completo: `com.ibm.cics.server.debug.Plugin`) proporciona una interfaz de finalidad general para registrar la implementación del plugin.

Estas interfaces se proporcionan en `com.ibm.cics.server.jar`, y se documentan en Javadoc (consulte "La biblioteca de clases de Java para CICS (JCICS)" en la página 53 para obtener más información).

El fragmento de código que aparece en Figura 9 muestra una implementación de ejemplo de la interfaz DebugControl.

```
public interface DebugControl
{
    // Se llama antes de invocar un método de objeto de aplicación o un principal de programa.
    public void startDebug(java.lang.String className,java.lang.String methodName);

    // Se llama después de invocar un método de objeto de aplicación o un principal de programa.
    public void stopDebug(java.lang.String className,java.lang.String methodName);

    // Se llama antes de suprimir un objeto de aplicación.
    public void exitDebug();

}
public interface Plugin
{
    // Inicializador, se llama cuando se registra el plugin.
    public void init();
}
```

Figura 9. Definiciones de las interfaces DebugControl y Plugin

El fragmento de código que aparece en Figura 10 muestra una implementación de ejemplo de las interfaces DebugControl y Plugin.

```
import com.ibm.cics.server.debug.*;

public class SampleCICSDebugPlugin
    implements Plugin, DebugControl
{
    // Implementación del inicializador de plugin.
    public void init()
    {
        // Este método se llama cuando el middleware Java de CICS carga y
        // registra el plugin. Puede utilizarse para realizar cualquier inicialización
        // necesaria para la implementación del control de depuración.
    }

    // Implementaciones de los métodos de control de depuración.
    public void startDebug(java.lang.String className,java.lang.String methodName)
    {
        // Este método se llama inmediatamente antes de invocar el método
        // de aplicación. Se puede utilizar para lanzar la operación de una herramienta de depuración. JCICS
        // las llamadas, como Task.getTask, se pueden utilizar aquí para obtener más
        // información sobre la aplicación.
    }

    public void stopDebug(java.lang.String className,java.lang.String methodName)
    {
        // Este método se llama inmediatamente después de invocar el método
        // de aplicación. Se puede utilizar para suspender la operación de una herramienta de depuración.
    }

    public void exitDebug()
    {
        // Este método se llama inmediatamente antes de suprimir el objeto
        // de aplicación. Se puede utilizar para terminar la operación de una herramienta de depuración.
    }
}
```

Figura 10. Implementaciones de ejemplo de las interfaces DebugControl y Plugin

Para activar una implementación de plugin de depuración, defina una o más de las siguientes propiedades del sistema en el archivo de propiedades de JVM correspondiente a la JVM:

Plugin de depuración de contenedor EJB
Si se define la siguiente propiedad del sistema, el código Java registra el plugin proporcionado en la capa del servidor EJB de CICS cuando se inicializa el contenedor EJB.

```
-Dcom.ibm.cics.server.debug.EJBPlugin=<nombre de clase completo,
   por ejemplo com.ibm.cics.server.debug.SampleCICSDebugPlugin>
```

Plugin de depuración de CORBA
Si se define la siguiente propiedad del sistema, el código Java registra el plugin proporcionado en el ORB de CICS cuando se inicializa el ORB.

```
-Dcom.ibm.cics.server.debug.CORBAPlugin=<nombre de clase completo,
   por ejemplo com.ibm.cics.server.debug.SampleCICSDebugPlugin>
```

Plugin de depuración de Java de CICS
Si se define la siguiente propiedad del sistema, un código Java adicional registra el plugin proporcionado en el derivador de CICS cuando se ejecuta el programa Java.

```
-Dcom.ibm.cics.server.debug.WrapperPlugin=<nombre de clase completo,
   por ejemplo com.ibm.cics.server.debug.SampleCICSDebugPlugin>
```

Puede desencadenarse más de una interfaz de plugin cuando se ejecuta una aplicación Java. Por ejemplo, si se registran implementaciones de plugin para las tres interfaces y se ejecuta un método de enterprise bean, los plugins de derivador JCICS, CORBA, y EJB se desencadenan sucesivamente.

Capítulo 9. Tecnologías estables de Java

CORBA, IIOP y Enterprise beans son tecnologías de Java que son estables en CICS. No utilice estas tecnologías para desarrollar nuevas aplicaciones.

Objetos CORBA sin estado

Desde la perspectiva del cliente, un objeto CORBA sin estado invocado mediante el ORB de CICS es solo una colección de métodos: es decir, un objeto sin estado.

Cada método remoto representa un fragmento de lógica que puede hacer una o más llamadas de API de CICS (incluidas llamadas a enlaces de programa) a programas de CICS existentes. Los objetos CORBA sin estado de CICS se ejecutan en una JVM de CICS. Al finalizar el método remoto, el estado del objeto ya no está disponible.

Al igual que sucede con todos los programas Java que se ejecutan en una JVM continua en CICS, cualquier estado estático creado por un objeto CORBA persiste en la JVM para su recuperación posterior en una tarea. Sin embargo, no hay afinidad entre un cliente de CORBA y una JVM de CICS, por lo que no existe la certeza de que dos solicitudes de CORBA posteriores que utilicen el mismo socket se procesen en la misma JVM (ni en la misma región CICS). Esto significa que la disponibilidad de un estado estático inicializado previamente no se puede dar por seguro.

Por lo tanto, en todo método remoto debe incluirse suficiente información en su lista de parámetros para permitir que complete su trabajo. No se pasa información al ORB del servidor mediante la referencia a objeto, excepto el tipo de objeto, que se utiliza para encontrar la clase de implementación. Sin embargo, los métodos del objeto pueden guardar el estado en almacenamiento de datos gestionado por aplicación entre invocaciones. Tendrán que asegurarse de que se pasa suficiente información como parámetros a los métodos posteriores, de manera que se puede recuperar el estado guardado.

Un objeto CORBA puede hacer llamadas de IIOP salientes, incluidas llamadas a enterprise beans que se ejecuten en el mismo CorbaServer o en uno distinto. Un objeto CORBA incluso puede pasar una referencia a sí mismo en un método de IIOP remoto. Esto se conoce como una **referencia de devolución de llamada**. Sin embargo, si el objeto de destino utiliza la referencia de devolución de llamada para llamar al primer objeto CORBA, esta nueva solicitud se procesa en una nueva JVM; por lo tanto, no tiene acceso a ningún estado de la JVM original.

Las invocaciones de método pueden participar en **transacciones distribuidas** del servicio de transacción de objetos (OTS). Si un cliente llama a una aplicación IIOP dentro del ámbito de una **transacción del OTS**, la información sobre dicha transacción del OTS fluye como un parámetro extra en la llamada a IIOP. Si un objeto de destino CORBA sin estado implementa CosTransactions::TransactionalObject, dicho objeto se trata como transaccional.

Desarrollo de objetos CORBA sin estado

Los objetos CORBA sin estado son aplicaciones de servidor Java que se comunican con una aplicación cliente utilizando el protocolo IIOP. No se mantiene ningún

estado en los atributos del objeto entre invocaciones del cliente sucesivas de métodos remotos; el estado se inicializa al principio de cada llamada a método remoto y es referenciado mediante parámetros explícitos.

Nota: Por un *método remoto* nos referimos a un método que se pueda llamar desde un cliente remoto. Es decir, un método público que se expone como parte de una de las interfaces remotas (que pueden ser muchas) del objeto que se declaran en el IDL para el objeto, y no un método interno al que no se puede acceder desde un cliente remoto.

En el modelo de programación del servidor, cada método es una subrutina. Los parámetros que se pasan le permiten establecer el estado temporal de cualquier base de datos o aplicación existente para realizar lógica empresarial, para almacenar datos en las bases de datos o aplicaciones existentes, para devolver resultados cuando los devuelve la subrutina o para generar una excepción. Los métodos remotos de un objeto CORBA sin estado —es decir, aquellos que puede llamar un cliente remoto— pueden llamarse entre sí localmente o llamar a métodos no remotos sin que se pierda el estado temporal del objeto. El estado temporal solo se descarta al final de la solicitud del método remoto iniciado por el cliente, cuando se envía la respuesta a la solicitud del cliente.

Puede desarrollar una aplicación CORBA sin estado utilizando cualquiera de dos métodos distintos:

1. Utilice el estilo de desarrollo CORBA típico, mediante el que se define una interfaz de aplicación en Interface Definition Language (IDL) y luego la aplicación se codifica para esa interfaz. Este método se describe en las secciones siguientes.
2. Utilice el estilo de desarrollo de Java típico, en el que se desarrolla una aplicación de invocación a método remoto (RMI) y después existe la opción de generar el IDL. Este enfoque se conoce como invocación de método remoto sobre protocolo Internet InterORB (RMI-IIOP) y se describe en "Desarrollo de una aplicación CORBA sin estado RMI-IIOP" en la página 221.

Para desarrollar un objeto CORBA sin estado utilizando el primer método (estilo CORBA), tiene que realizar los siguientes pasos:

1. Utilice el Interface Definition Language (IDL) para definir las interfaces y las operaciones del objeto.
2. Ejecute el compilador IDL a Java (IDLJ) con el IDL para generar clases de apéndice (stub) y esqueleto (skeleton) para el objeto.
3. Escriba una aplicación cliente que realice llamadas al servidor utilizando la clase de apéndice generada.
4. Escriba una aplicación de servidor (el objeto CORBA sin estado) que amplíe la clase de esqueleto base generada.
5. Compile y empaquete las aplicaciones cliente y de servidor.
6. Defina los recursos de CICS para el servidor, y añada el archivo JAR de la aplicación de servidor a la vía de acceso de clase estándar en el perfil de JVM para la JVM que utiliza la aplicación.

Para desarrollar un objeto CORBA sin estado utilizando el segundo método (estilo Java), tiene que realizar los siguientes pasos:

1. Escriba una interfaz remota para la aplicación de servidor (el objeto CORBA sin estado).

2. Escriba una aplicación cliente que realice llamadas al servidor utilizando esta interfaz remota.
3. Escriba una aplicación de servidor que implemente la interfaz remota.
4. Compile las aplicaciones cliente y de servidor.
5. Ejecute el compilador RMI (RMIC) de Java con la interfaz remota y la aplicación de servidor para generar clases de apéndice y de enlace (tie) para el objeto.
6. Empaquete las aplicaciones cliente y de servidor.
7. Defina los recursos de CICS para el servidor, y añada el archivo JAR de la aplicación de servidor a la vía de acceso de clase estándar en el perfil de JVM para la JVM que utiliza la aplicación.
8. Como opción, cree IDL para la aplicación para su uso por parte de clientes CORBA que no sean Java.

Existen beneficios e inconvenientes para cada uno de los dos métodos. Una de las diferencias principales es que el método CORBA necesita que el objeto CORBA sin estado amplíe una clase base generada. Dado que Java solo soporta una jerarquía de herencia única, esto significa que no puede hacer que su objeto CORBA sin estado amplíe una clase que elija. El método RMI-IIOP le permite utilizar una jerarquía de herencia que elija para el objeto CORBA sin estado, ya que el objeto únicamente tiene que implementar una interfaz específica.

Los nombres de la interfaz y la operación CORBA se correlacionan con implementaciones Java correspondientes. Puede desarrollar implementaciones de servidor que utilicen las clases Java de JCICS (JCICS) para acceder a servicios de CICS. Consulte la *Referencia de clases JCICS* para obtener detalles de las clases JCICS y "Programación Java utilizando JCICS" en la página 53 para obtener una explicación de cómo desarrollar aplicaciones de servidor mediante ellas.

Las clases JCICS están completamente documentadas en JAVADOC.html, que se genera desde las definiciones de clases. Este está disponible a través del Information Center de CICS, en la *Referencia de clases JCICS*.

Obtención de una Interoperable Object Reference (IOR)

Para localizar un objeto de servidor en tiempo de ejecución, la aplicación cliente necesita una referencia al mismo.

Esta referencia se denomina **Interoperable Object Reference** (**IOR**) (referencia de objeto interoperable). Una IOR es una serie de texto codificada de una manera específica, de forma que un ORB de cliente pueda descodificar la IOR para localizar el objeto de servidor remoto. Contiene suficiente información para permitir:

- Dirigir una solicitud al servidor correcto (host, número de puerto).
- Localizar o crear un objeto (nombre de clase, datos de instancia).

Las IOR pueden devolverlas los métodos de servidor, pero es necesaria una clase factory para crear una IOR inicial. CICS utiliza la clase GenericFactory de CORBA LifeCycle Services (CosLifeCycle) para este fin. Una aplicación cliente puede utilizar esta GenericFactory para crear las IOR para cada objeto CORBA sin estado en el tiempo de ejecución. Sin embargo, la GenericFactory en sí es un objeto CORBA sin estado, por lo que la aplicación cliente necesitará *su propia* IOR antes de poder crear la IOR del objeto de destino.

Utilice el mandato PERFORM CORBASERVER PUBLISH para publicar una IOR mediante series para la clase GenericFactory. Entonces se crea la IOR GenericFactory, se almacena en el directorio de estantería (un directorio de z/OS UNIX asociado con el CorbaServer) y se publica en el servidor de nombres. La aplicación cliente puede utilizar la IOR de GenericFactory para crear las IOR para cualquier objeto CORBA sin estado que exista para este CorbaServer (y únicamente para dicho CorbaServer). La IOR se publica con el nombre `genfac.ior`. La manera en la que el cliente localiza la IOR GenericFactory en el tiempo de ejecución es una decisión de la arquitectura de aplicaciones. La IOR se podría recuperar de una ubicación bien conocida de un espacio de nombres de JNDI o conservarse de forma local en la máquina cliente, o algún otro proceso podría acceder a ella.

Para publicar la IOR, puede utilizar el mandato **CEMT PERFORM CORBASERVER**, o puede emitir un mandato **EXEC CICS PERFORM CORBASERVER** desde una aplicación de CICS.

El archivo `genfac.ior` se graba en el directorio de estantería de CORBASERVER:
/directorio_estantería/id_aplicación/corbaserver/

donde:

directorio_estantería
: Es el nombre del directorio de estantería (SHELF) especificado en la definición de recurso de CORBASERVER, cuyo valor predeterminado es `/var/cicsts/`

id_aplicación
: Es el identificador APPLID asociado con la región CICS.

corbaserver
: Es el nombre del recurso CORBASERVER.

Puede descargar la IOR a su estación de trabajo cliente (en modalidad ASCII) desde el directorio de estantería mediante FTP. Como alternativa, el cliente puede utilizar la interfaz de JNDI para obtener la IOR del servidor de nombres.

Debido a la naturaleza sin estado del objeto, no tiene prácticamente ningún sentido que un cliente cree más de una instancia de una clase. Una vez que un cliente ha creado una instancia de un objeto, por ejemplo, `bankaccountfacilitator`, el mismo objeto se puede utilizar para acceder a la cuenta del Sr. X y a la cuenta del Sr. Y; el número de cuenta es un parámetro de entrada en cada método.

Nota: Hemos llamado al objeto de este ejemplo bankaccountfacilitator (facilitador de cuentas) para que pueda realizar acciones en cada cuenta. Si lo hubiéramos llamado bankaccount (cuenta bancaria) podría implicar que la instancia siempre representaba a la cuenta del Sr. X.

Creación del Interface Definition Language (IDL)

Si utiliza el estilo de desarrollo CORBA para crear una aplicación de objeto CORBA sin estado, debe crear un archivo de IDL de OMG que contenga las definiciones de interfaces que la implementación del servidor soportará.

Nota: Esta sección asume que se utiliza el estilo de desarrollo CORBA para crear una aplicación de objeto CORBA sin estado (método 1 en “Desarrollo de objetos CORBA sin estado” en la página 211, en lugar del método RMI-IIOP). El método RMI-IIOP se describe en “Desarrollo de una aplicación CORBA sin estado RMI-IIOP” en la página 221.

Un archivo de IDL de OMG describe los tipos de datos, las operaciones y los objetos que los clientes pueden utilizar y que un servidor debe proporcionar para una implementación de un objeto determinado.

Para obtener información acerca de cómo escribir IDL, consulte la publicación de OMG, *Common Object Broker: Architecture and Specification*, que se puede obtener en el sitio web de OMG en http://www.omg.org/.

Las definiciones de IDL se procesan con un compilador IDL a Java (que a veces también se denomina "analizador" o "generador"). Debe utilizar un compilador provisto por el entorno del servidor para generar clases de esqueletos (skeleton) y ayudante (helper) del lado del servidor y un compilador provisto por el entorno del cliente para generar clases de apéndice (stub, llamado a veces "proxy") y auxiliares del lado de cliente. Las clases de esqueleto adecuadas para su uso con CICS se pueden crear utilizando el compilador IDLJ provisto con cualquier IBM Java 2 SDK. Si se utiliza un compilador IDLJ que no sea IBM, la clase de esqueleto resultante puede ser adecuada o no para su uso con CICS. Si tiene dudas, puede utilizar el compilador IDLJ que se entrega con el Java SDK provisto en z/OS que utiliza CICS.

Las clases de apéndice o de proxy producidas por el compilador IDL de IBM (IDLJ) son adecuadas para su uso con cualquier ORB de IBM. Si utiliza un ORB del lado de cliente de un proveedor distinto, debe utilizar el compilador IDL provisto con dicho ORB. Si utiliza clases de apéndice generadas para el ORB de un proveedor con el ORB de otro proveedor, los resultados son indefinidos: los apéndices pueden funcionar o no.

Los proxys y los esqueletos proporcionan la información específica del objeto necesaria para que un ORB distribuya una invocación de método.

La Figura 11 en la página 216 muestra cómo el mismo archivo de IDL se utiliza para generar distintas clases utilizadas por el cliente y el servidor.

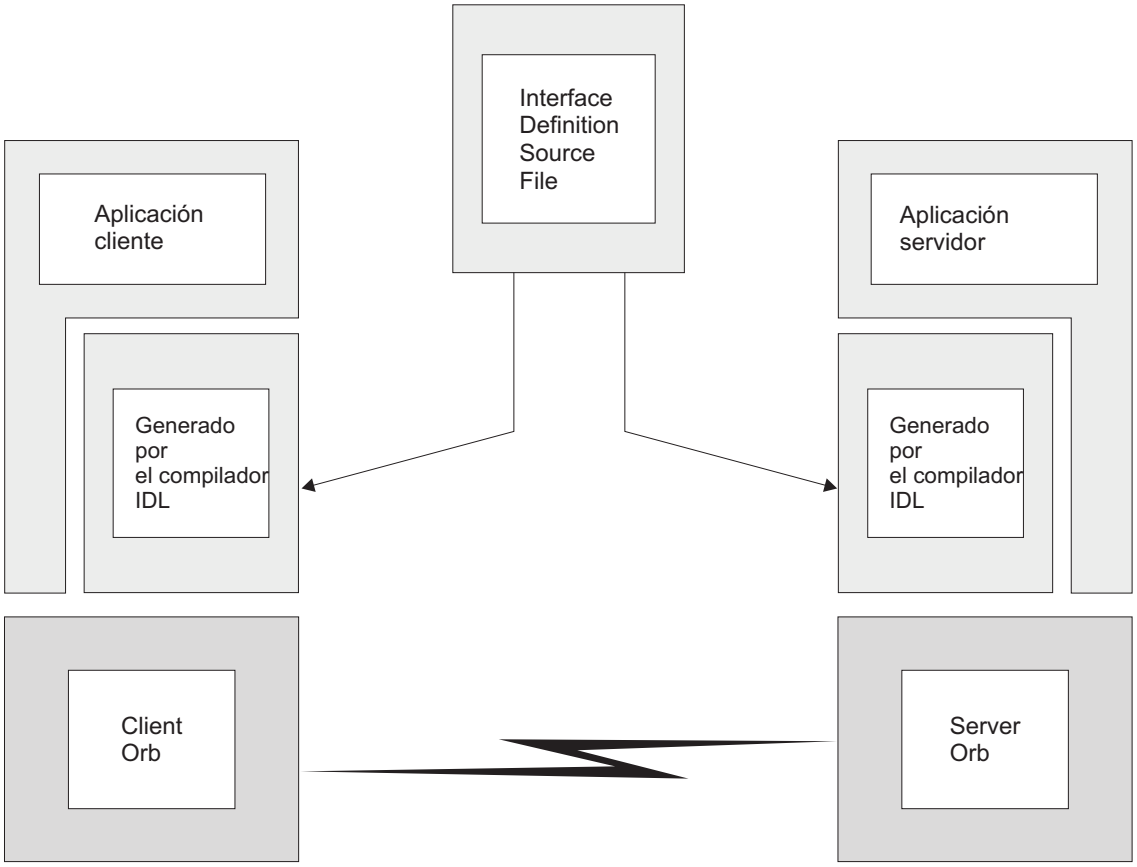


Figura 11. IDL y código generado

Desarrollo de un programa de servidor IIOP

El programa de servidor se puede desarrollar en cualquier plataforma que soporte Java. Por ejemplo, una estación de trabajo NT, AIX o el entorno UNIX System Services de z/OS.

Acerca de esta tarea

Nota: Esta sección asume que se utiliza el estilo de desarrollo de CORBA para crear una aplicación de objeto CORBA sin estado (método 1 en "Desarrollo de objetos CORBA sin estado" en la página 211, en lugar del método RMI-IIOP). El método RMI-IIOP se describe en "Desarrollo de una aplicación CORBA sin estado RMI-IIOP" en la página 221.

Son necesarios los siguientes pasos:

Procedimiento

1. Escriba la definición IDL de las interfaces y las operaciones que forman la aplicación.
2. Compile el archivo IDL para generar clases de esqueleto y ayudante de CORBA utilizando el mandato **idlj** del compilador IDL, que forma parte del Java 2 SDK.

   Nota:

a. Debe utilizarse un compilador IDL a Java proporcionado por IBM para hacerlo. El compilador IDL a Java proporcionado con la versión Sun del Java 2 SDK puede no ser 100% compatible con el ORB de IBM.
b. El mandato **idlj** no se proporciona como parte del Java Runtime Environment (JRE); necesitará instalar un SDK completo en la máquina para que funcione.

El compilador IDL se puede invocar de las siguientes formas:

```
idlj [opciones] <archivo_idl>
```

Donde <archivo _idl> es el nombre del archivo que contiene las definiciones de IDL, y [opciones] es una combinación de las siguientes opciones, que pueden aparecer en cualquier orden. <archivo_idl> es necesario y debe aparecer al final. Como mínimo debe especificarse **-f**.

Por ejemplo:

```
idlj -v -fall myidl.idl
```

También debe especificar la opción **-oldImplBase** para asegurar que se genera una implementación compatible con CICS. Si no utiliza esta opción, la implementación generada utilizará el adaptador de objetos portátiles (POA), que no está soportado en CICS. Por ejemplo:

```
idlj -v -fall -oldImplBase myidl.idl
```

-d<símbolo>
El equivalente de la siguiente línea en un archivo IDL: `#define <símbolo>`

-emitAll
Emita todos los tipos, incluidos los encontrados en archivos `#included`.

-f<lado>
Defina los enlaces que emitir. <lado> puede ser:

client No aplicable para CICS.

server No genera suficientes clases para un uso normal.

all emite todos los enlaces.

serverTIE
no soportado en CICS.

allTIE no soportado en CICS.

Si no se especifica esta opción, se asume **-fclient**. En la mayoría de los casos se debe utilizar **-fall**.

-i<include path>
Añada otro directorio. De forma predeterminada, se explora el directorio actual en busca de archivo de inclusión.

-keep Si un archivo que se va a generar ya existe, no se sobrescribe. De manera predeterminada, se sobrescribe.

-oldImplBase
Esta opción es necesaria. Si se omite, IDLJ genera código que utiliza el adaptador de objetos portátiles (POA). El POA no está soportado en CICS.

-pkgPrefix <t> <pkg>
Asegúrese de que, siempre que se encuentre el tipo o módulo <t>, este resida en <pkg> en todos los archivos generados. <t> es un nombre estilo Java completo.

-v Modalidad detallada.

3. Escriba la implementación de servidor en código Java. El compilador IDL generará una clase abstracta llamada *_nombre_interfaz***ImplBase**. Su programa debe ampliarla. Si se van a crear objetos de este tipo mediante la Generic Factory, la clase de implementación debe llamarse *_nombre_interfaz*Impl. Si no se utiliza este convenio de denominación, GenericFactory no podrá crear referencias a su objeto CORBA. Por ejemplo:

   ```
   public class _BankAccountImpl extends _BankAccountImplBase
   ```

   Su clase de implementación puede utilizar la API de JCICS para interaccionar con los servicios CICS tradicionales.
4. Compile el programa y la salida desde el paso 2, utilizando el compilador javac o un equivalente, como VisualAge for Java. Asegúrese de que la ubicación de los archivos de salida se añade al final de la vía de acceso de clase estándar de CICS, mediante la opción CLASSPATH_SUFFIX en el perfil de JVM.

Ejemplo

Este ejemplo describe una cuenta bancaria cuyo contenido puede consultarse y actualizarse. El ejemplo tiene un parámetro que identifica la instancia de BankAccount para cumplir una restricción de 'stateless' (sin estado). El siguiente IDL define la interfaz y las operaciones:

```
module bank {

// esta interfaz se utiliza para gestionar la interfaz de cuentas
bancarias BankAccount {
  exception ACCOUNT_ERROR { long errcode; string message;};

  // métodos de consulta
  long querybalance(in long acnum) raises (ACCOUNT_ERROR);
  string queryname(in long acnum) raises (ACCOUNT_ERROR);
  string queryaddress(in long acnum) raises (ACCOUNT_ERROR);

  // métodos de establecimiento
  void setbalance(in long acnum, in long balance) raises (ACCOUNT_ERROR);
  void setaddress(in long acnum, in string address) raises (ACCOUNT_ERROR);
};
};
```

La implementación de servidor del IDL anterior debe llamarse `_BankAccountImpl` si se van a crear objetos de este tipo mediante la GenericFactory y debe ampliar `_BankAccountImplBase`, que se genera mediante el compilador IDL. Es parte del paquete de Java `bank`. Puede ver detalles completos de esta implementación en la aplicación de ejemplo BankAccount CORBA sin estado distribuida en :

```
/usr/lpp/cicsts/<nombre_usuario>/samples/dfjcorb
```

donde *nombre_usuario* es un nombre que se puede elegir durante la instalación de CICS y que de forma predeterminada es `cicsts42`.

Para utilizar este ejemplo, necesita los siguientes recursos:

- Un recurso TCPIPSERVICE definido e instalado para escuchar en un puerto determinado en CICS. Este TCPIPSERVICE debe:
  - Definirse para utilizar el protocolo Inter-ORB de Internet.

  - Estar en estado “abierto” para recibir solicitudes.
- Un recurso CORBASERVER definido para procesar solicitudes IIOP en TCPIPSERVICE.

Como opción, puede elegir añadir una definición REQUESTMODEL, con el fin de forzar que la solicitud se procese con un TRANSID determinado.

Desarrollo del programa cliente IIOP

Se escribe una aplicación cliente que realiza llamadas al servidor utilizando la clase de apéndice generada.

Acerca de esta tarea

Nota: Esta sección asume que se utiliza el estilo de desarrollo CORBA para crear una aplicación de objeto CORBA sin estado (método 1 en “Desarrollo de objetos CORBA sin estado” en la página 211, en lugar del método RMI-IIOP). El método RMI-IIOP se describe en “Desarrollo de una aplicación CORBA sin estado RMI-IIOP” en la página 221.

Procedimiento

1. Procese el archivo IDL con un compilador IDL a Java adecuado para su sistema cliente (utilizando el mismo archivo IDL que utilizara para compilar la aplicación cliente).
2. Obtenga una referencia de objeto mediante series para GenericFactory descargando `genfac.ior` (en modalidad ASCII) del directorio de estantería de CorbaServer en el que se creó cuando se publicó el recurso CORBASERVER. Como alternativa, puede utilizar JNDI, cuando se publica unA IOR de Generic Factory para el CorbaServer en el espacio de nombres si se emite un mandato **`EXEC CICS PERFORM CORBASERVER PUBLISH`** o **`CEMT PERFORM CORBASERVER PUBLISH`**. Si tiene pensado utilizar JNDI, debe definir un servidor de nombres; consulte “Definición de servidores de nombres” en la página 395. La IOR se une al contexto identificado por el prefijo de JNDI en la definición de recurso CORBASERVER, con la misma GenericFactory. Por ejemplo, el nombre de vía de acceso sería:

   `/jndiprefix/GenericFactory`

   Consulte la *Guía de definición de recurso de CICS* y el manual *Transacciones suministradas de CICS*.
3. Escriba su programa cliente, que contenga llamadas al servidor. Para obtener una referencia de objeto inicial, utilice la GenericFactory como se muestra en “Ejemplo de cliente”.
4. Compile el programa cliente y la salida del paso 1, con javac o un compilador equivalente.

Resultados

Ejemplo de cliente

El siguiente ejemplo muestra cómo un programa cliente utiliza el servicio GenericFactory para crear un objeto **account** (cuenta). Primero el cliente debe crear un proxy para la GenericFactory.

Los enlaces Java para parte de los módulos CosLifeCycle y CosNaming de CORBA son necesarios. Si el ORB del cliente no los proporciona, puede compilarlos utilizando el compilador IDL a Java del ORB, desde el IDL de los servicios CORBA

disponibles desde el sitio web de la OMG (www.omg.org). Como alternativa, puede utilizar la versión Java precompliada del IDL que se incluye en
/usr/lpp/cicsts/<*cicsts42*>/lib/omgcos.jar

Donde *cicsts42* es el valor elegido para el parámetro de instalación USSDIR que definiera al instalar CICSTS.

El archivo JAR se debe descargar en modalidad binaria y hacer que esté disponible en la entrada del entorno CLASSPATH del cliente.

El siguiente ejemplo y los ejemplos proporcionados necesitan enlaces que se pueden importar como org.omg.CosNaming y org.omg.CosLifeCycle.

Para crear un objeto de cuenta, primero el cliente debe crear un proxy para la GenericFactory. El siguiente ejemplo asume que existe una referencia mediante series a la GenericFactory en un archivo disponible para un cliente y que la devuelve el método getFactoryIOR().

```
import java.io.*;
import org.omg.CORBA.*;
import org.omg.CosLifeCycle.*;
import org.omg.CosNaming.*;
public class bankLineModeClient{

//El siguiente método lee la IOR desde un archivo y lo devuelve en la serie
 String factoryIOR = getFactoryIOR();
// Convierta la referencia mediante serie en el proxy
 org.omg.CORBA.Object genFacRef = orb.string_to_object(factoryIOR);
// Reduzca a la interfaz correcta.
 GenericFactory fact = GenericFactoryHelper.narrow(genFacRef);
```

Ahora que el cliente tiene un fábrica genérica, puede utilizarla para crear un objeto cuenta.

```
// La fábrica genérica necesita una clave, que es una secuencia de componentes de
nombre (namecomponent).
 NameComponent nc = new NameComponent("bank::BankAccount","object interface");
 NameComponent key[] = {nc};
//La fábrica genérica también necesita criterios (que ignora).
 NVP mycriteria[] = {};

//Ahora cree el objeto
 org.omg.CORBA.Object objRef = fact.create_object(clave, mis_criterios);
// y reduzca a la interfaz correcta.
BankAccount acctRef = BankAccountHelper.narrow(objRef);
```

Ahora que el cliente tiene un objeto, puede utilizarlo:

```
int ac1 = 1234; // Cuenta de Tony
int ac2 = 3456; // Cuenta de Lou
String name;
String address;
int balance;


try {
  name=acctRef.queryname(ac1);
  System.out.println("a/c num:"+ac1+" name:"+name);
}
catch (exception e) {
  System.err.println("error de consulta");
}
```

Nota: Un NVP (par de valor de nombre) es un tipo de datos definido en el IDL de CORBA para la interfaz de la fábrica genérica.

Desarrollo de una aplicación CORBA sin estado RMI-IIOP

Puede utilizar el estilo de desarrollo de invocación de método remoto sobre protocolo Internet InterORB (RMI-IIOP) para crear una aplicación de objeto CORBA sin estado.

Este es el estilo de desarrollo definido en el método 2 de "Desarrollo de objetos CORBA sin estado" en la página 211, en lugar del método de desarrollo CORBA descrito en secciones anteriores.

El método RMI-IIOP implica el desarrollo de una aplicación de invocación a método remoto (RMI) Java estándar y desplegarla para utilizar IIOP como su protocolo de transporte. Se trata del método que utilizan los enterprise beans.

Nota: Esta sección documenta específicamente cómo desarrollar una aplicación CORBA sin estado utilizando RMI-IIOP. Enterprise beans se despliegan con otras herramientas. Para obtener información sobre cómo desplegar enterprise beans, consulte "Despliegue de enterprise beans" en la página 321.

Al utilizar RMI-IIOP no es necesario definir una interfaz utilizando IDL, aunque, si es necesario, el IDL se puede generar más tarde. En su lugar, comenzamos por definir al menos una interfaz remota. En este contexto, una "interfaz remota" significa cualquier interfaz Java que amplíe `java.rmi.Remote`. Esto no es lo mismo que la "interfaz remota" de un enterprise bean. Si utilizamos la terminología que acabamos de definir, tanto la interfaz remota como la interfaz inicial de un enterprise bean se considerarían "interfaces remotas", porque en último término amplían `java.rmi.Remote`.

Esta interfaz remota se debe codificar para seguir las reglas de la RMI Java. A continuación se muestra un ejemplo de interfaz remota:

```
package hello;
public interface HelloWorldRMI extends java.rmi.Remote
{
   public String sayHello(String msgFromClient) throws java.rmi.RemoteException;
}
```

La interfaz anterior define un único método llamado sayHello que toma una serie (String) como parámetro y devuelve una serie. Todos los métodos de la interfaz se deben definir para generar java.rmi.RemoteException.

A continuación debe proporcionar una implementación de lado del servidor de esta interfaz. A continuación se muestra un ejemplo:

```
package hello;
public class _HelloWorldRMIImpl implements HelloWorldRMI
{
 public String sayHello(String msgFromClient)
 { return "Hola: ha dicho: " + msgFromClient;}
}
```

La clase de implementación implementa la interfaz creada anteriormente. El convenio de denominación utilizado para la clase de implementación es _<*nombre_de_interfaz*>Impl. Este convenio de denominación es necesario si el objeto de servidor se va a ubicar mediante el método de fábrica genérica CosLifeCycle de CORBA. Si no se utiliza este convenio de denominación, la fábrica genérica no podrá crear instancias del objeto CORBA sin estado.

Una de las ventajas de RMI-IIOP sobre el proceso de desarrollo más tradicional basado en IDL es que no se ve obligado a ampliar una clase base. Esto significa que puede elegir utilizar su propia jerarquía de herencia si lo desea. También puede implementar varias interfaces remotas con un único objeto de servidor.

Debe compilar ambas clases anteriores utilizando el compilador `javac` o uno equivalente.

Lo siguiente que hay que hacer es producir el archivo Tie (intercambio de información técnica) del lado del servidor para este objeto CORBA sin estado. Esto se realiza mediante el compilador RMI (RMIC). Debe utilizar un compilador RMI incluido con un IBM Java 2 SDK. Si se utiliza la versión del RMIC proporcionada con un Java 2 SDK, no está garantizado que el archivo Tie generado funcione con el ORB de CICS.

El mandato que utilizar es el siguiente:

```
rmic -iiop hello._HelloWorldRMIImpl
```

Tenga en cuenta que RMIC se ejecuta en la clase de implementación del lado del servidor.

A continuación necesitamos la clase de apéndice del lado de cliente. Esto también se produce mediante el compilador RMI. Asegúrese de utilizar un compilador RMI adecuado para su ORB de cliente. El mandato que utilizar es el siguiente:

```
rmic -iiop hello.HelloWorldRMI
```

Tenga en cuenta que RMIC se ejecuta en la clase de interfaz remota.

Una vez completada, debe tener disponibles las siguientes clases:

```
hello\HelloWorldRMI.class              - la interfaz remota
hello\_HelloWorldRMIImpl.class         - el objeto CORBA sin estado
hello\__HelloWorldRMIImpl_Tie.class    - el archivo Tie RMI-IIOP del lado del servidor
hello\_HelloWorldRMI_Stub.class        - archivo de resguardo RMI-IIOP del cliente
```

Lo siguiente que hay que hacer es escribir la aplicación cliente. La aplicación cliente es muy similar a la aplicación cliente desarrollada utilizando el método basado en IDL para el desarrollo CORBA (descrito en “Desarrollo del programa cliente IIOP” en la página 219). Como antes, aún tenemos que encontrar una referencia al objeto CORBA sin estado mediante la fábrica genérica CosLifeCycle de CORBA. Aquí hay parte de un ejemplo de aplicación cliente RMI-IIOP:

```
ORB orb = ORB.init((String[]) null, (java.util.Properties) null);

// El siguiente método lee la IOR de la fábrica genérica desde un archivo y lo devuelve
// en la serie.
String factoryIOR = getFactoryIOR();

// Convierta la referencia mediante serie en el proxy.
org.omg.CORBA.Object genFacRef = orb.string_to_object(factoryIOR);

// Reduzca a la interfaz correcta.
GenericFactory fact = GenericFactoryHelper.narrow(genFacRef);

// La fábrica genérica necesita una clave, que es una secuencia de componentes de
nombre (namecomponent).
NameComponent nc = new NameComponent("hello::HelloWorldRMI","object interface");

//Ahora cree el objeto
org.omg.CORBA.Object objRef=fact.create_object(new NameComponent[]{nc},
                                               new NVP[] {});
```

```
// y reduzca a la interfaz correcta mediante la operación reducida RMI-IIOP.
HelloWorldRMI remote = (HelloWorldRMI) javax.rmi.PortableRemoteObject.narrow
                       (objRef, HelloWorldRMI.class);

// Invoque el método remoto.
System.out.println("Recibido del servidor: "+remote.sayHello("¡Hola!")+"\n");}
```

Al igual que sucede con la aplicación cliente basada en IDL, será necesario tener el archivo `omgcos.jar` del directorio CICS lib de z/OS UNIX en las máquinas de estación de trabajo y de cliente para encontrar las clases CosLifeCycle.

Todo lo que queda por hacer es empaquetar las aplicaciones del lado del servidor y del cliente en archivos JAR y añadir el archivo JAR del lado del servidor a la vía de acceso de clase estándar.

Si se desea generar IDL, para la interfaz remota RMI-IIOP, que sea adecuado para utilizar con una aplicación cliente CORBA no basada en Java, utilice el siguiente mandato:

```
rmic -idl hello.HelloWorldRMI
```

Aplicaciones cliente CORBA autónomas de CICS

El soporte para CORBA de CICS se centra principalmente en el soporte de objetos del lado del servidor IIOP: es decir, enterprise beans y objetos CORBA sin estado. Estos componentes del lado del servidor se ejecutan en un servidor EJB/CORBA de CICS, en un entorno de ejecución de CorbaServer, representado por un recurso CORBASERVER. Como se ejecutan en un servidor EJB/CORBA de CICS, tienen acceso a un amplio conjunto de características de ORB.

En esta sección, el término "*aplicaciones cliente CORBA autónomas de CICS*" se refiere a aplicaciones de CICS que:

1. Son aplicaciones cliente CORBA.
2. Se definen para CICS como aplicaciones Java estándar, mediante una definición PROGRAM en la que se especifica JVM=YES.
3. Crean una instancia de ORB utilizando el operador `new`.
4. No se ejecutan en un entorno de ejecución CorbaServer de CICS.

Las aplicaciones cliente CORBA autónomas de CICS no se ejecutan en un servidor EJB/CORBA de CICS, por lo que no tienen acceso a la misma calidad de soporte de CORBA que los componentes del lado del servidor. El ORB disponible para estas aplicaciones cliente es un ORB solo de cliente al que a veces se llama "JCICS ORB". Este ORB no puede escuchar en un socket las conexiones de entrada; por lo tanto, cualquier IOR publicada por este ORB no se soporta. De forma parecida, una aplicación cliente CORBA de CICS no puede lanzar una transacción del OTS distribuida (ni participar en ella). Una aplicación cliente CORBA de CICS tampoco puede participar en autenticación de identidad con afirmación.

Estas limitaciones no se amplían al entorno del servidor de CICS. Cualquier objeto de servidor de un servidor EJB/CORBA de CICS puede realizar llamadas salientes IIOP de cliente que participen en una transacción del OTS, siempre que la instancia de ORB utilizada para realizar las llamadas salientes sea el ORB actual del servidor EJB/CORBA de CICS. Si el servidor crea una nueva instancia de ORB utilizando el operador new, CICS no puede propagar automáticamente la transacción existente y el contexto de seguridad mediante este nuevo ORB. Un objeto de servidor de IIOP puede obtener, mediante programación, un descriptor de contexto para la instancia actual del ORB del servidor utilizando la siguiente llamada a método estático:

```
com.ibm.cics.iiop.ORBFactory.getORB()
```

Interoperatividad CORBA

La implementación de CICS de la arquitectura CORBA proporciona un enlace entre aplicaciones basada en ORB CORBA y en servicios CICS, lo que incluye enterprise beans.

Un enterprise bean alojado por CICS se puede hacer que interaccione con objetos de otras regiones CICS (incluidas regiones CICS de nivel anterior, desde CICS TS 1.3 en adelante), WebSphere Application Server y servidores de aplicaciones y ORB J2EE de terceros. Los enterprise beans están disponibles para clientes CORBA puros y pueden actuar como clientes para objetos CORBA remotos (potencialmente implementados en un lenguaje de programación distinto y alojados en una plataforma distinta).

El ORB CICS se puede utilizar solo para albergar aplicaciones de cliente y de servidor escritas en Java. Sin embargo, se puede utilizar para interaccionar con ORB remotos que atiendan a clientes y servidores escritos en otros lenguajes de programación.

Utilización de clientes CORBA que no sean Java

Distintos lenguajes de programación necesitan distintos enlaces de lenguaje para un ORB.

Esto necesita un nivel de interoperatividad entre los ORB que se debe de tener en cuenta. La arquitectura CORBA define los enlaces de lenguaje para varios lenguajes, entre los que se incluyen C++, Java, COBOL, Ada, PL/I, Smalltalk y otros. Tenga en cuenta que los enlaces de lenguaje para lenguajes de programación no soportan todas las característica de IDL y de IIOP. En concreto, los tipos de valor se han definido solo para los enlaces de C++ y Java. El acceso de CORBA a enterprise beans requiere tipos de valor, por lo que actualmente solo las aplicaciones C++ y Java pueden acceder a la mayoría de los enterprise beans mediante una interfaz CORBA.

Escritura de un cliente CORBA en un enterprise bean

En el caso de lenguajes de programación de cliente distintos a Java, como C++, la arquitectura CORBA a menudo es la única opción viable para acceder a enterprise beans.

Acerca de esta tarea

En el caso de lenguajes de programación de cliente distintos a Java, como C++, la arquitectura CORBA a menudo es la única opción viable para acceder a enterprise beans. Los enterprise beans están disponibles para clientes CORBA mediante el modelo de programación CORBA, del siguiente modo:

- Escriba un enterprise bean.
- Genere IDL para el enterprise bean mediante el compilador RMI con la opción -IDL. (Esto es lo inverso al modelo CORBA habitual, en el que se utiliza IDL para generar el objeto).

  Si se utilizan solo primitivas de CORBA como tipos de datos y de retorno, será más fácil acceder al bean desde clientes que no sean Java.
- Mediante un compilador IDL adecuado para el entorno del cliente, compile el IDL para generar apéndices del lado de cliente.
- Escriba el cliente utilizando el apéndice generado.

- Haga que para la aplicación cliente haya disponible una IOR para el enterprise bean. La IOR contiene suficiente información para que cualquier ORB de CORBA localice el enterprise bean.

Incluso si un bean de sesión se ha codificado para utilizar solo primitivas CORBA como tipos de parámetro y de retorno, los tipos de excepción se siguen devolviendo como tipos de valor CORBA. Si el ORB del cliente CORBA no soporta tipos de valor, se verá obligado a trabajar con excepciones desconocidas.

Nota: No se recomienda utilizar un cliente CORBA de Java para un enterprise bean. Utilice RMI-IIOP en su lugar.

Enterprise beans como clientes de CORBA

Los enterprise beans son objetos de Java que operan en un entorno de ejecución sofisticado que incluye un ORB.

Si el enterprise bean va a realizar llamadas de IIOP salientes a objetos CORBA remotos (sin utilizar RMI-IIOP), se recomienda encarecidamente que la aplicación utilice la instancia de ORB existente. Si el enterprise bean crea una nueva instancia de ORB utilizando el operador new, CICS no puede propagar a solicitudes de método en este nuevo ORB ni la transacción existente ni el contexto de seguridad en el que el bean se ejecuta.

Si necesita obtener un descriptor de contexto para el ORB actual en un enterprise bean, puede utilizar la siguiente llamada a método estática:

```
com.ibm.cics.iiop.ORBFactory.getORB()
```

Conjuntos de códigos

CICS puede aceptar tipos de datos char/wchar y string/wstring de GIOP solo si se codifican utilizando una de estas páginas de códigos.

- UCS2: el conjunto de códigos estándar de Java (Unicode)
- UTF-8

Utilización de las aplicaciones de ejemplo de IIOP

Estas aplicaciones de ejemplo enseñan a utilizar aplicaciones IIOP (objetos CORBA sin estado) y el soporte de programación CICS Java (JCICS).

Aplicación de ejemplo HelloWorld

Esta aplicación de ejemplo proporciona una prueba muy simple de los componentes IIOP. El programa cliente:

- Lee el archivo genfac.ior para obtener una referencia a la fábrica genérica.
- Utiliza la fábrica genérica para crear un objeto HelloWorld.
- Invoca el método sayHello para enviar un saludo al servidor ("Hello from HelloWorldClient", Hola desde el cliente HelloWorld) y recibir un saludo de este como respuesta ("Hello from CICS TS", Hola desde CICS TS).

El diseño de la aplicación se describe en los comentarios del código.

Aplicación de ejemplo BankAccount

Esta aplicación de ejemplo consta de los siguientes componentes principales:

1. Una aplicación CICS tradicional que utiliza BMS y la API **EXEC CICS**, escrita en C. Esta aplicación consta de de dos transacciones:

   BNKI Inicializa un archivo con información sobre distintas cuentas bancarias. Estas cuentas tienen números que van del 23 al 30.

BNKQ
Consulta la información de las cuentas. También hay un programa de CICS, DFH$IICC, que realiza una comprobación de crédito sobre una cuenta.

2. Una implementación de una interfaz IDL que define un objeto de cuenta bancaria. La implementación está escrita en Java y se ejecuta como un objeto CORBA sin estado. Esta implementación utiliza un archivo de cuentas bancarias para acceder a la información de cuenta y el programa de comprobación de crédito DFH$IICC para obtener calificaciones de crédito.
3. Una aplicación cliente de CORBA escrita en Java que visualiza información sobre objetos de cuenta bancaria.

El diseño de la aplicación se describe en los comentarios del código.

Configuración del entorno de ejemplo IIOP

Puede utilizar los programas de ejemplo incluidos para configurar un entorno IIOP en CICS.

Antes de empezar

Para configurar CICS como un servidor o un cliente IIOP, debe contar con una región CICS que tenga permiso para acceder a z/OS UNIX y al IBM 64 bits SDK para z/OS, Java Technology Edition. Language Environment debe estar configurado y activo.

Procedimiento

1. Defina el parámetro de JCL **REGION** en la secuencia de trabajos de inicio para una región CICS que soporte IIOP. Defina el valor a un mínimo de `1000M`.
2. Defina los parámetros de inicialización del sistema en el trabajo de inicio para una región CICS que soporte IIOP:

   ```
   EDSALIM=500M
   MAXJVMTCBS=número
   TCPIP=YES
   ```

   Defina un valor mínimo de `500M` para el parámetro **EDSALIM**. Para encontrar un valor adecuado para el parámetro **MAXJVMTCBS** , consulte “Gestión de la agrupación de JVM para mejorar el rendimiento” en la página 172.
3. Añada sentencias DD a la secuencia de trabajos de inicio para una región CICS que soporte IIOP para crear estos archivos:

   DFHEJDIR
   Un archivo compartido recuperable que contiene el directorio de secuencias de solicitud. Este puede ser un archivo VSAM o una tabla de datos de recurso de acoplamiento. CICS proporciona JCL de ejemplo para ayudarle a crear este archivo, en el miembro DFHDEFDS de la biblioteca SDFHINST.

   DFHEJOS
   Un archivo compartido no recuperable utilizado por CICS cuando se instalan recursos de CORBASERVER. Este archivo también se utiliza para almacenar beans de sesión con estado que se hayan desactivado. DFHEJOS puede ser un archivo VSAM o una tabla de datos de recurso de acoplamiento. CICS proporciona JCL de ejemplo para ayudarle a crear este archivo, en el miembro DFHDEFDS de la biblioteca SDFHINST.

Las definiciones de conjunto de datos VSAM locales de ejemplo para estos archivos se suministran en el grupo de RDO DFHEJVS proporcionado por CICS. Estos conjuntos de datos deben autorizarse con RACF para el acceso de UPDATE. Consulte Authorizing access to CICS data sets, en la *Guía de seguridad RACF de CICS*.

4. Cree un directorio de estantería en z/OS UNIX y concédale al ID de usuario de la región CICS acceso completo al mismo. Consulte Concesión a las regiones CICS de acceso a z/OS UNIX System Services para obtener orientación.
5. Elija un perfil de JVM adecuado y asegúrese de que CICS puede localizar dicho perfil, como se describe en "Configuración de JVM en agrupación" en la página 95.
6. Asegúrese de que el valor de JAVA_HOME está definido correctamente en el perfil de JVM para la aplicación del lado del servidor. JAVA_HOME define el directorio de instalación para el IBM 64 bits SDK para z/OS, Java Technology Edition. El valor predeterminado para la versión 6.0.1 del SDK es `/usr/lpp/java/J6.0.1_64/`.
7. Asegúrese de que se añaden los siguientes archivos a una vía de acceso de clases adecuada en el perfil de JVM:
   - Los archivos de origen y los archivos make de Java de ejemplo se almacenan en el sistema de archivos de z/OS UNIX System Services durante la instalación de CICS, en los siguientes directorios:
     - `$CICS_HOME/samples/dfjcorb/HelloWorld`
     - `$CICS_HOME/samples/dfjcorb/BankAccount`
   - La ubicación en la que se han compilado las clases para las aplicaciones del lado del servidor.

   Para obtener orientación, consulte "Clases y vías de acceso de clases en las JVM" en la página 9.
8. Asegúrese de que están instalados los grupos de definición de recurso DFHIIOP y DFH$IIOP incluidos con CICS. El grupo DFH$IIOP incluido contiene las siguientes definiciones:
   - Las definiciones de recurso necesarias para la región de escucha TCP/IP (que también puede ser la misma región que ejecuta los programas de ejemplo):
     - Definición SSL TCPIPSERVICE
     - Definición NOSSL TCPIPSERVICE
   - Definiciones de recurso necesarias para el programa de ejemplo HelloWorld:
     - Definición IIHE TRANSACTION
     - Definición DFJIIRH REQUESTMODEL
     - Definición IIOP CORBASERVER
   - Definiciones de recurso necesarias para el programa de ejemplo BankAccount:
     - Definición DFH$IIBI PROGRAM
     - Definición DFH$IIBQ PROGRAM
     - Definición DFH$IICC PROGRAM
     - Definición BANKINQ MAPSET
     - Definición BNKI TRANSACTION
     - Definición BNKQ TRANSACTION
     - Definición BNKS TRANSACTION
     - Definición BANKACCT FILE
     - Definición DFJIIRB REQUESTMODEL

– Definición IIOP CORBASERVER

Las definiciones TCPIPSERVICE e IIOP CORBASERVER hacen referencia a los números de puerto predeterminados, 683 y 684. Tal vez tenga que cambiarlos a los números de puerto que tenga disponibles. Asimismo, la definición IIOP hace referencia a CICSHOST como el host del CorbaServer. Debe cambiarlo a su propio nombre de host. Consulte Recursos TCPIPSERVICE y Recursos CORBASERVER.

9. Convierta y compile los siguientes programas y el conjunto de correlaciones de lenguaje C de CICS e inclúyalos en una biblioteca en la concatenación DFHRPL de CICS. Se almacenan en SDFHSAMP durante la instalación de CICS. El orden de compilación es importante. Tanto DFH$IIBI como DFH$IICC se pueden compilar independientemente, pero el conjunto de correlaciones BMS DFH$IIMA se debe compilar antes de compilar DFH$IIBQ. Para obtener orientación sobre la conversión, la compilación y el enlace de programas de aplicación de CICS, consulte la *Guía de programación de la aplicación de CICS*.

   El archivo DFH$IIMA contiene un conjunto de correlaciones BANKINQ con dos correlaciones. Compile y enlace el conjunto de correlaciones BANKINQ.

   Consulte Installing map sets and partition set, en la *Guía de programación de la aplicación de CICS*, para obtener orientación sobre cómo compilar y enlazar correlaciones BMS.

   DFH$IIBI
   Programa de C que inicializa el archivo BANKACCT. Ejecute mediante la transacción BNKI.

   DFH$IIBQ
   Programa de C que consulta las cuentas mantenidas en BANKACCT.

   DFH$IICC
   Programa de C que realiza una comprobación de crédito. Es llamado por DFH$IIBQ.

   DFH$IIMA
   Conjunto de correlaciones BMS BANKINQ.

   Nota: En los nombres de los programas de ejemplos y los archivos descritos en este libro, el símbolo del dólar ($) se utiliza como símbolo de moneda nacional y se supone que se le asignará el punto de código X'5B' de EBCDIC. En determinados países, se asigna un símbolo de moneda diferente, por ejemplo, el símbolo de la libra (£) o el yen (¥), al mismo punto de código EBCDIC. En estos países, se debe utilizar el símbolo de moneda adecuado en lugar del símbolo del dólar.

10. Para compilar el cliente HelloWorld de IIOP son necesarias las clases de tiempo de ejecución CosLifeCycle y CosNaming. Si el entorno de su ORB de cliente no proporciona estos servicios ya compilados, puede utilizar el archivo `omgcos.jar` incluido en el directorio `$CICS_HOME/lib`. Como alternativa, puede optar por compilar estas clases a partir del IDL original proporcionada por OMG. En este caso, hay disponible una copia de los archivos IDL relevantes en `$CICS_HOME/samples/dfjcorb`. El proceso de convertir el IDL puro en código ejecutable depende de cada ORB, pero si utiliza un ORB proporcionado con una JVM, es probable que los siguientes mandatos funcionen:

```
idlj -pkgprefix CosNaming org.omg -pkgprefix CosLifeCycle org.omg -fall CosLifeCycle.idl
idlj -pkgprefix CosNaming org.omg -pkgprefix CosLifeCycle org.omg -fall CosNaming.idl
javac org\omg\CosLifeCycle\*.java org\omg\CosNaming\NamingContextPackage\*.java
      org\omg\CosNaming\*.java
```

Debe asegurarse de que estas clases están disponibles en su variable de entorno classpath al intentar compilar cualquier aplicación cliente CORBA sin estado de CICS.

11. Obtenga un archivo `genfac.ior` que contenga una referencia de objeto con la fábrica genérica del servidor y colóquelo en el directorio actual. El archivo `genfac.ior` se crea cuando se emite un mandato **PERFORM CORBASERVER PUBLISH** para el recurso IIOP CORBASERVER de ejemplo instalado. Se escribe en el directorio de estantería del CORBASERVER:

    `/var/cicsts/id_aplicación/IIOP`

    donde *id_aplicación* es el identificador APPLID asociado con la región CICS.

    Para publicar la definición CORBASERVER, puede utilizar un mandato **CEMT PERFORM CORBASERVER**, o un mandato **EXEC CICS PERFORM CORBASERVER** emitido desde una aplicación de CICS.

    Puede descargar la IOR a su estación de trabajo cliente (en modalidad ASCII) desde el directorio de estantería mediante FTP.

Resultados

Ha utilizado los programas de ejemplo para configurar un entorno IIOP.

Ejecución de la aplicación de ejemplo HelloWorld de IIOP

Esta sección indica qué debe hacer para ejecutar la aplicación de ejemplo HelloWorld.

Abarca los temas siguientes:

- "Creación de la aplicación HelloWorld del lado del servidor"
- "Creación de la aplicación HelloWorld del lado del cliente"
- "Ejecución de la aplicación de ejemplo HelloWorld" en la página 230

Creación de la aplicación HelloWorld del lado del servidor:

El archivo make de `$CICS_HOME/samples/dfjcorb/HelloWorld/server` crea todo lo necesario para la aplicación del lado del servidor.

Debe añadirse `$CICS_HOME/samples/dfjcorb/HelloWorld/server` a la vía de acceso de clase estándar mediante la opción CLASSPATH_SUFFIX en el perfil de JVM, DFHJVMCD.

Para crear los programas, especifique el mandato `make` desde `$CICS_HOME/samples/dfjcorb/HelloWorld/server`. Este mandato crea el objeto HelloWorld.

Creación de la aplicación HelloWorld del lado del cliente:

`$CICS_HOME/samples/dfjcorb/HelloWorld/client` contiene la parte de cliente CORBA de la aplicación. El archivo de origen de la aplicación de cliente Java se llama `HelloWorldClient.java`. Esta aplicación se puede ejecutar con cualquier ORB compatible con CORBA.

Acerca de esta tarea

Son necesarios los siguientes pasos para compilar la aplicación de cliente Java:

1. Descargue los siguientes archivos a la estación de trabajo cliente (en modalidad ASCII):

- .../dfjcorb/HelloWorld/HelloWorld.idl
- .../dfjcorb/HelloWorld/client/HelloWorldClient.java

2. Compile el IDL provisto con el compilador IDL a Java del ORB del cliente para producir los apéndices (stubs) del lado del cliente Java que necesita la aplicación de ejemplo. Estos apéndices se crearán en un subdirectorio llamado hello. Mueva la aplicación cliente HelloWorldClient.java a este subdirectorio.
3. Compile la aplicación cliente, asegurándose de que las clases Java producidas en el paso anterior están disponibles en la variable de entorno CLASSPATH. Para compilar la aplicación cliente desde el directorio actual, especifique:

   ```
   javac hello\HelloWorldClient.java
   ```

   También necesita las clases de tiempo de ejecución CosLifeCycle y CosNaming. Si el entorno de su ORB de cliente no proporciona estos servicios ya compilados, puede utilizar el archivo omgcos.jar incluido en el directorio $CICS_HOME/lib en z/OS UNIX. Como alternativa, puede optar por compilar estas clases a partir del IDL original proporcionada por OMG. En este caso, hay disponible una copia de los archivos IDL relevantes en $CICS.HOME/samples/dfjcorb/.

   El proceso de convertir el IDL puro en código ejecutable depende de cada ORB, pero si utiliza un ORB proporcionado con una JVM, es probable que los siguientes mandatos funcionen:

```
idlj -pkgprefix CosNaming org.omg -pkgprefix CosLifeCycle org.omg -fall CosLifeCycle.idl
idlj -pkgprefix CosNaming org.omg -pkgprefix CosLifeCycle org.omg -fall CosNaming.idl
javac org\omg\CosLifeCycle\*.java
              org\omg\CosNaming\NamingContextPackage\*.java
              org\omg\CosNaming\*.java
```

   Estas clases deben estar disponibles en la vía de acceso de clases al intentar crear cualquier aplicación cliente CORBA sin estado de CICS.

Ejecución de la aplicación de ejemplo HelloWorld:

Debe utilizar este mandato para ejecutar la aplicación cliente.

Acerca de esta tarea

```
java hello.HelloWorldClient
```

Ejecución de la aplicación de ejemplo BankAccount de IIOP

Esta sección indica qué debe hacer para ejecutar la aplicación de ejemplo BankAccount.

Trata los temas siguientes:

- "Creación de la aplicación BankAccount del lado del servidor" en la página 231
- "Creación de la aplicación BankAccount del lado del cliente" en la página 231
- "Ejecución de la aplicación de ejemplo BankAccount" en la página 231

Creación del archivo VSAM:

Debe definir el archivo VSAM que mantenga los datos de cuenta bancaria utilizando estos parámetros de IDCAMS.

```
DEFINE CLUSTER (                        -
                NAME (CICS610.BANKACCT )     -
                CYLINDERS(01)                -
                REUSE                        -
                KEYS(4 0)                    -
                RECORDSIZE(168 168))
```

Creación de la aplicación BankAccount del lado del servidor:

El archivo make de `$CICS_HOME/samples/dfjcorb/BankAccount/server` crea todo lo necesario para la parte CORBA de la aplicación del lado del servidor.

Añada `$CICS_HOME/samples/dfjcorb/BankAccount/server` a la vía de acceso de clase estándar mediante la opción CLASSPATH_SUFFIX en el perfil de JVM, DFHJVMCD.

Para crear el programa de servidor Java, especifique el mandato `make` desde `$CICS_HOME/samples/dfjcorb/BankAccount/server`.

Creación de la aplicación BankAccount del lado del cliente:

`$CICS_HOME/samples/dfjcorb/BankAccount/javaclient` contiene la parte del cliente CORBA de la aplicación. El archivo de origen de la aplicación de cliente Java se llama `bankLineModeClient.java`. Esta aplicación se puede ejecutar con cualquier ORB compatible con CORBA.

Acerca de esta tarea

Son necesarios los siguientes pasos para crear la aplicación de cliente Java:

1. Descargue los siguientes archivos a la estación de trabajo cliente (en modalidad ASCII):
   - `.../dfjcorb/BankAccount/BankAccount.idl`
   - `.../dfjcorb/BankAccount/javaclient/bankLineModeClient.java`
2. Compile el IDL provisto con el compilador IDL a Java del ORB del cliente para producir los apéndices (stubs) del lado del cliente Java que necesita la aplicación de ejemplo. Tras compilar el IDL para crear el subdirectorio, `bank`, mueva el archivo Java a este subdirectorio. Compile el programa desde el directorio actual utilizando el siguiente mandato:

   ```
   javac bank\bankLineModeClient.java
   ```
3. Asegúrese de que las clases Java producidas en el paso anterior están disponibles en la variable de entorno CLASSPATH.

   También necesita las clases de tiempo de ejecución CosLifeCycle y CosNaming. Si el entorno de su ORB de cliente no proporciona estos servicios ya compilados, puede obtenerlos del mismo modo que en “Creación de la aplicación HelloWorld del lado del cliente” en la página 229.

Ejecución de la aplicación de ejemplo BankAccount:

Debe realizar estos pasos para ejecutar la aplicación de ejemplo:

Acerca de esta tarea

Procedimiento

1. Ejecute la transacción BNKI de CICS para cargar datos en el archivo de cuenta.
2. Ejecute la aplicación cliente mediante:

   ```
   java  bank.bankLineModeClient
   ```

Utilización de enterprise beans

Esta sección indica lo que necesita saber para desarrollar y utilizar enterprise beans en CICS.

¿Qué son los enterprise beans?

CICS soporta la arquitectura Enterprise JavaBeans (EJB).

Si necesita una descripción completa de la arquitectura EJB, consulte http://www.oracle.com/technetwork/java/index.html.

La sección trata los temas siguientes:

Enterprise beans

La *Enterprise JavaBeans Specification, versión 1.1* define un modelo para el desarrollo de componentes de servidor de Java reutilizables conocidos como *enterprise beans*. Estos componentes se pueden utilizar en cualquier servidor de aplicaciones que brinde los servicios y las interfaces que define dicha especificación.

Puede configurar CICS como un *servidor EJB*. CICS brinda un entorno de ejecución en el que las solicitudes de servicios de EJB se correlacionan con servicios de CICS existentes o ampliados.

Puede escribir enterprise beans que brinden a los clientes Java acceso a su inversión pasada en aplicaciones y datos de CICS. Por ejemplo, puede escribir enterprise beans que:

- Utilicen las clases de JCICS para acceder a recursos de CICS.Los enterprise beans que utilizan las clases de JCICS no se pueden llevar a un entorno que no sea CICS.
- Utilicen JCICS para enlazar con programas de CICS existentes escritos en lenguajes de procedimientos, como COBOL.

La Figura 12 muestra, de forma simplificada, un servidor de aplicaciones EJB de CICS que interacciona con su entorno. Muestra cómo enterprise beans que se han desarrollado en una estación de trabajo se instalan en un servidor EJB mediante un proceso conocido como *despliegue*. Una vez instalados en un servidor, los enterprise beans se ejecutan en una máquina virtual Java (JVM) cuando lo solicita un programa cliente.

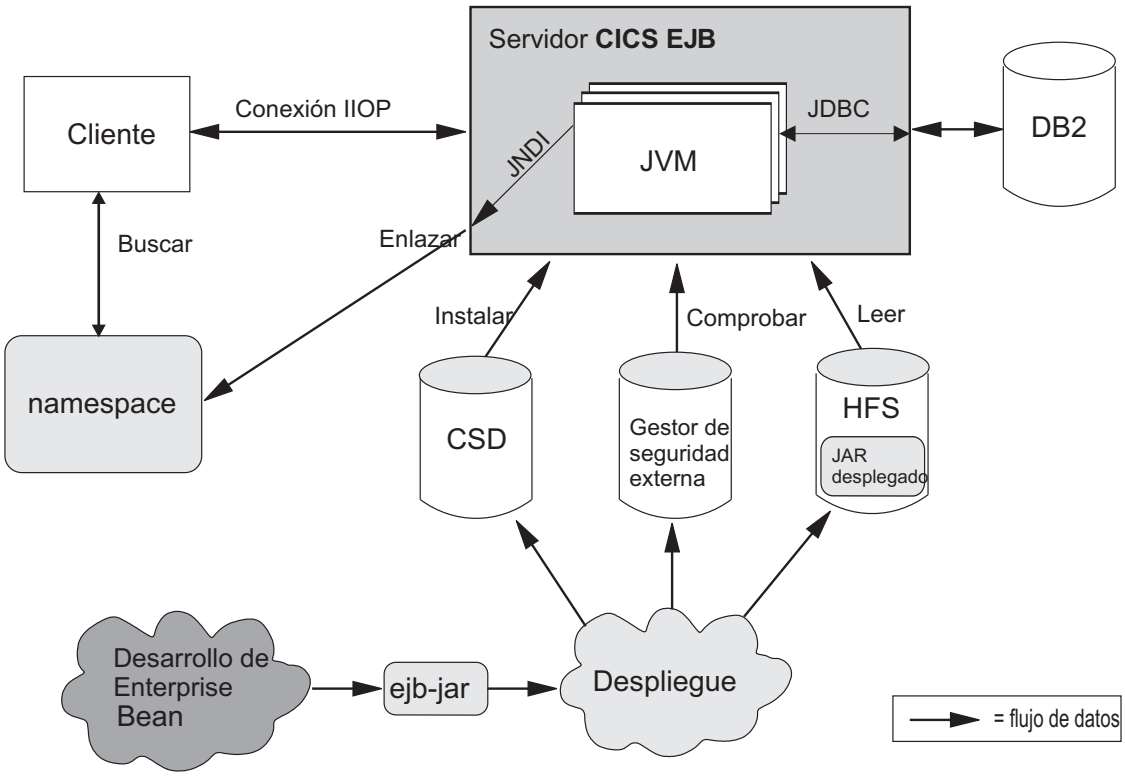


Figura 12. Un servidor de aplicaciones EJB de CICS

JavaBeans y Enterprise JavaBeans

JavaBeans y Enterprise JavaBeans son arquitecturas de componente para el lenguaje Java.

Componentes:

Un **componente** es un bloque de creación de software reutilizable; un fragmento ya creado de código de aplicación encapsulado que se puede combinar con otros componentes y con código escrito manualmente para producir una aplicación personalizada rápidamente.

Un desarrollador de aplicaciones puede aprovechar un componente sin tener que acceder a su código fuente. Los componentes se pueden personalizar para ajustarse a los requisitos específicos de una aplicación mediante un conjunto de valores de propiedad externos. Por ejemplo, un componente de botón tiene una propiedad que especifica el título que debe aparecer en el botón. Un componente de gestión de cuenta tiene una propiedad que especifica la ubicación de la base de datos de cuentas.

Los componentes se ejecutan en una construcción llamada **contenedor**, que, entre otras cosas, proporciona un proceso de sistema operativo en el que ejecutar el componente.

El **modelo de componente** define las interfaces mediante las que el componente interacciona con su contenedor y con otros componentes. El desarrollador de un componente puede codificarlo empleando distintos métodos internos y propiedades, pero para asegurarse de que se puede utilizar con otros componentes, debe implementar las interfaces definidas en el modelo de componente. Estas interfaces también permiten cargar componentes en herramientas RAD (desarrollo rápido de aplicaciones) como WebSphere Studio Application Developer.

JavaBeans:

Un **JavaBean** es un componente de software autocontenido y reutilizable, escrito en Java, y generalmente concebido para utilizarse en una aplicación *de escritorio o cliente*.

Generalmente, los JavaBeans de escritorio cuentan con un elemento visual y se ejecutan en el mismo tipo de contenedor visual, como un formulario, un panel o una página web. Los ejemplos pueden ir desde un botón simple hasta un reproductor de CD de software completo.

Los desarrolladores de beans pueden utilizar una herramienta visual, como WebSphere Studio Application Developer, para crear JavaBeans. Los desarrolladores de aplicaciones pueden utilizar dichas herramientas para "conectar" JavaBeans entre sí y crear una aplicación mayor, y también para definir las propiedades de beans individuales.

Enterprise JavaBeans:

La **arquitectura de Enterprise JavaBeans** soporta los *componentes de servidor*. Los componentes de servidor son componente de la aplicación que se ejecutan en un servidor de aplicaciones como CICS. A diferencia de los componentes de escritorio, no cuentan con un elemento visual y el contenedor en el que se ejecutan no es visual.

Los componentes de servidor que se escriben según la especificación Enterprise JavaBeans se conocen como **enterprise beans**. Pueden transportarse a cualquier servidor de aplicaciones compatible con EJB.

Para ser útiles, los componentes de servidor necesitan acceder a los servicios de infraestructura del servidor de aplicaciones, como su servicio de comunicación distribuida, servicios de directorio y denominación, servicio de gestión de transacciones, servicios de acceso a datos y persistencia y servicios de compartición de recursos. Distintos servidores de aplicaciones implementan estos servicios de infraestructura utilizando distintas tecnologías. Sin embargo, un servidor de

aplicaciones compatible con EJB proporciona un enterprise bean con acceso a estos servicios mediante interfaces estándar y gestiona muchas de ellas en nombre del bean.

Los desarrolladores de beans pueden utilizar una herramienta visual, como WebSphere Studio Application Developer, para crear enterprise beans. Los desarrolladores de aplicaciones pueden combinar llamadas a método con JavaBeans de escritorio, servlets web y código escrito a mano para formar aplicaciones cliente/servidor.

El servidor EJB: visión general

Un servidor de aplicaciones compatible con EJB se conoce como *servidor EJB*.

Un servidor EJB podría ser un supervisor de proceso de transacción como CICS, un servidor web, una base de datos o cualquier otro tipo de servidor. Tenga en cuenta que un servidor EJB de CICS puede contar con varias regiones CICS, como se describe en "Servidores lógicos: enterprise beans en un sysplex" en la página 250.

Un servidor EJB proporciona un conjunto estándar de servicios para soportar componentes de enterprise bean. Estos servicios incluyen:

- Soporte de la interfaz de la invocación a método remoto (RMI) de Java que utilizan los enterprise beans para comunicación. La RMI tiene dos opciones de protocolo de transporte: JRMP para interoperación de Java a Java e IIOP para interoperación entre lenguajes, mediante el uso de un intermediario para solicitudes de objetos (ORB) de CORBA. (Para obtener una descripción del ORB de CICS, consulte "El intermediario para solicitudes de objetos (ORB)" en la página 382).

  CICS Transaction Server para z/OS, Versión 4 Release 2 soporta RMI sobre IIOP (RMI-IIOP), pero no JRMP. (JRMP es un protocolo patentado que no se puede utilizar para interoperar con componentes que no sean Java. CICS no soporta las transacciones distribuidas mediante JRMP).
- Un contenedor, llamado **contenedor EJB**, que proporciona servicios de gestión para enterprise beans.
- Un servicio de gestión de transacción distribuida que implementa la interfaz javax.transaction.UserTransaction de la API de transacción Java (JTA). La interfaz javax.transaction.UserTransaction la utilizan los beans de sesión para gestionar sus propias transacciones.
- Servicios de seguridad.
- Soporte para la Java Naming and Directory Interface (JNDI). La API de JNDI proporciona funcionalidad de directorios y denominación para aplicaciones Java. Permite que un cliente localice un enterprise bean.
- Soporte para la interfaz de Java Data Base Connectivity (JDBC).

El contenedor EJB: visión general

Mientras que los JavaBeans de escritorio generalmente se ejecutan en un contenedor visual, como un formulario o una página web, un enterprise bean se ejecuta en un contenedor proporcionado por el servidor de aplicaciones.

El contenedor EJB crea y gestiona instancias de enterprise bean en tiempo de ejecución y proporciona los servicios que necesita cada enterprise bean que se ejecuta en él.

El contenedor EJB soporta un número de servicios implícitos, incluidos ciclo de vida, gestión de estado, seguridad y gestión de transacciones:

Ciclo de vida
Los enterprise beans individuales no necesitan gestionar la asignación de procesos, la gestión de hebras, la activación de objetos o la desactivación de objetos de forma explícita. El contenedor EJB automáticamente gestiona el ciclo de vida del objeto en nombre del enterprise bean.

Gestión de estado
Los enterprise beans individuales no necesitan guardar ni restaurar el estado de objeto entre llamadas a método de forma explícita. El contenedor EJB automáticamente gestiona el estado de objeto en nombre del enterprise bean.

Seguridad
Los enterprise beans individuales no necesitan autenticar usuarios ni comprobar los niveles de autorización de forma explícita. El contenedor EJB puede realizar automáticamente todas las comprobaciones de seguridad en nombre del enterprise bean.

Gestión de transacciones
Los enterprise beans individuales no necesitan especificar el código de demarcación de transacción para participar en transacciones distribuidas. El contenedor EJB puede gestionar automáticamente el inicio, la inscripción, la confirmación y la retrotracción de las transacciones en nombre del enterprise bean.

El entorno de ejecución:

Antes de poder desplegar enterprise beans en un servidor EJB, es necesario configurar su entorno de ejecución.

En CICS, esto se logra instalando la definición de recurso CORBASERVER. CORBASERVER define un entorno de ejecución para los enterprise beans y los objetos sin estado CORBA. Por comodidad, denominaremos al entorno de ejecución definido por una definición CORBASERVER como un **CorbaServer**.

Observe que:

- Un servidor EJB de CICS puede contener más de un CorbaServer.
- En un mismo CorbaServer puede desplegarse cualquier número de enterprise beans.
- Un enterprise bean específico se puede desplegar varias veces en el mismo servidor EJB de CICS, pero no en el mismo CorbaServer. (Dicho de otro modo, para instalar un enterprise bean específico varias veces en el mismo servidor EJB de CICS, es necesario instalarlo en distintos entornos de ejecución CorbaServer. Una razón para hacerlo podría ser hacer que el bean estuviera disponible con distintas propiedades de despliegue; consulte “Enterprise beans: el descriptor de despliegue” en la página 237). Cada despliegue resulta en la creación de un objeto inicial diferente (consulte “Enterprise beans: las interfaces inicial y de componente”).

Enterprise beans: las interfaces inicial y de componente

Las aplicaciones cliente no interaccionan con un enterprise bean directamente.

En su lugar, el cliente interacciona con el enterprise bean mediante dos objetos intermedios que el contenedor crea a partir de clases generadas por una herramienta de despliegue; una de estas clases implementa la **interfaz inicial** de

EJB y la otra la **interfaz de componente** de EJB. A medida que el cliente invoca operaciones utilizando estos objetos intermedios, el contenedor intercepta cada llamada a método e inserta los servicios de gestión.

Las interfaces inicial y de componente se implementan como objetos remotos de la RMI de Java, lo que permite que el ORB los soporte como objetos distribuidos.

La interfaz inicial
La interfaz inicial es el mecanismo mediante el que el cliente identifica el enterprise bean que desea. Permite a un cliente crear, eliminar y (en el caso de beans de entidad, no soportados por CICS) encontrar instancias existentes de enterprise beans. *Tenga en cuenta que el "cliente" puede no ser un programa que se ejecute en una estación de trabajo de una red; por ejemplo, podría ser un servlet que se ejecute en un servidor web, o un enterprise bean, un programa o un objeto en el servidor EJB local o en otro servidor EJB.*

Cuando se despliega un bean en un servidor EJB, el contenedor registra la interfaz inicial en un espacio de nombres que sea accesible de forma remota. Mediante la API de Java Naming and Directory Interface (JNDI), cualquier cliente con acceso al espacio de nombres puede localizar la interfaz inicial por su nombre. (Para ser precisos, el cliente localiza, por nombre, un objeto que implementa la interfaz inicial. La interfaz inicial amplía la interfaz EJBHome).

La interfaz de componente
La interfaz de componente permite que un cliente acceda a los métodos de negocio del enterprise bean. Intercepta todas las llamadas a método de negocio desde el cliente e inserta cualquier transacción, gestión de estado, persistencia y servicios de seguridad que se especificara cuando se desplegara el bean.

Cuando un cliente crea o encuentra una instancia de un enterprise bean, el contenedor devuelve un objeto de interfaz de componente (uno por instancia). (Para ser precisos, el contenedor devuelve una referencia a una instancia de una clase que implementa la interfaz de componente. La interfaz de componente amplía la interfaz EJBObject).

Enterprise beans: el descriptor de despliegue

Las reglas que regulan el ciclo de vida de un enterprise bean, la gestión de transacciones, la seguridad y la persistencia se definen en un documento XML asociado llamado **descriptor de despliegue**.

Consulte "Visión general del despliegue de enterprise beans" en la página 246.

Los componentes reutilizables pueden personalizarse mediante un conjunto de valores de propiedad externos, de forma que se puedan modificar para adecuarlos a los requisitos de una aplicación concreta sin cambiar el código fuente. Un desarrollador de enterprise beans puede proporcionar (en el descriptor de despliegue) un conjunto de **propiedades del entorno** que permitan al desarrollador de aplicaciones personalizar el bean. Por ejemplo, se puede utilizar una propiedad para especificar la ubicación de una base de datos o para especificar un idioma nacional predeterminado. En el tiempo de ejecución, se crea un objeto de entorno que contiene los valores de propiedad personalizados definidos durante el proceso de ensamblaje de aplicaciones o el proceso de despliegue de beans.

El servidor EJB: resumen

Este tema resume la información sobre los servidores EJB presentados en los temas anteriores.

La siguiente figura muestra objetos de enterprise bean en un servidor EJB de CICS.El contenedor EJB gestiona y proporciona servicios a los enterprise beans que contiene. Cuando se despliega un bean, la herramienta de despliegue genera las clases de interfaz EJB home (inicial) y component (componente).

La interfaz inicial es accesible mediante la JNDI e implementa servicios de ciclo vital para el bean. El cliente lo utiliza para crear, eliminar y (en el caso de beans de entidad no soportados directamente por CICS) encontrar instancias de enterprise beans.

El contenedor crea un objeto de interfaz de componente de EJB para cada instancia del bean. La interfaz de componente proporciona acceso a los métodos de negocio del bean. Intercepta todas las llamadas a método de negocio desde el cliente e implementa servicios de transacción, gestión de estado, persistencia y seguridad para el bean, en base a los valores del descriptor de despliegue del bean.

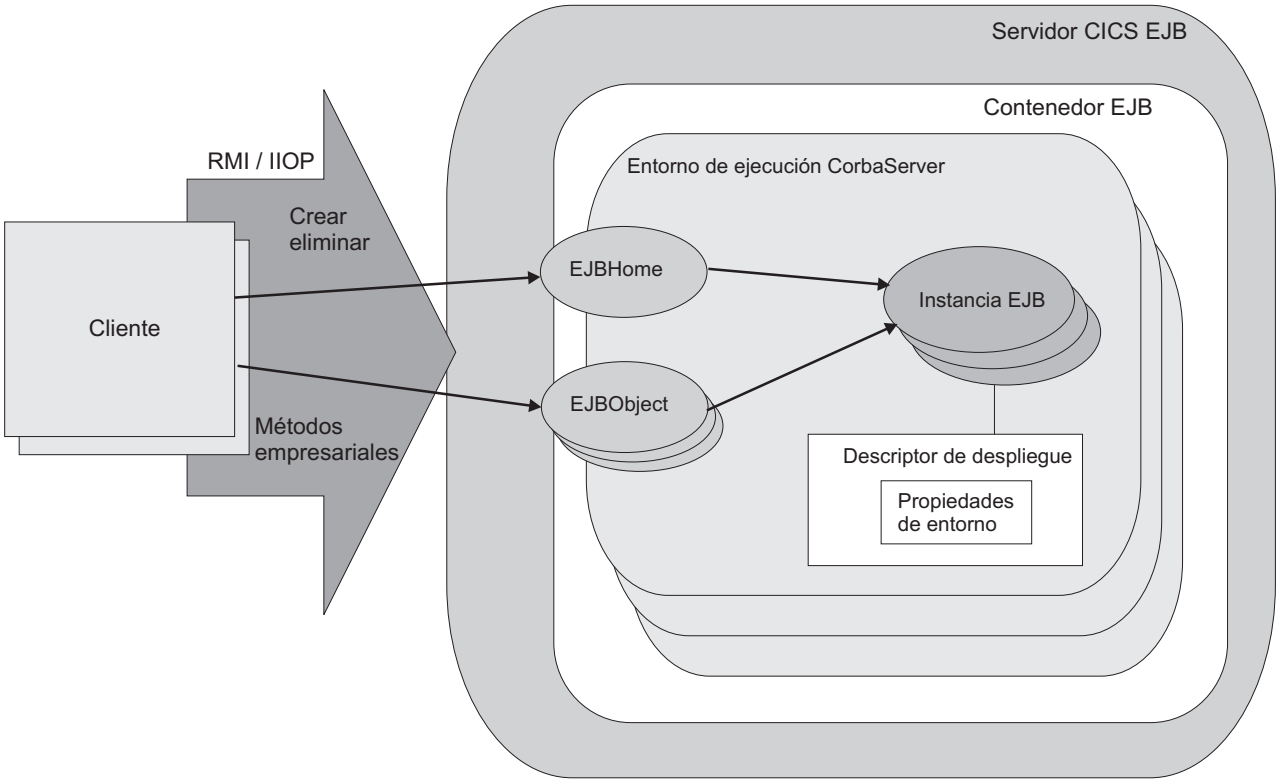


Figura 13. Objetos de enterprise bean de un servidor EJB de CICS

Tipos de enterprise bean

Esta sección analiza dos tipos de enterprise bean: **beans de sesión** y **beans de entidad**.

Beans de sesión: Un bean de sesión:

- Lo crea un cliente y representa una única conversación o sesión con dicho cliente.
- Generalmente, persiste solo durante la vida de la conversación con el cliente. En este sentido, se puede enlazar con una transacción pseudoconversacional.

  Si el desarrollador del bean elige guardar información más allá de la duración de una sesión, debe implementar operaciones de persistencia —por ejemplo, llamadas a JDBC o SQL— directamente en los métodos de clase del bean.

- Generalmente realiza operaciones en datos empresariales en nombre del cliente, como acceder a una base de datos o realizar cálculos.
- Puede ser transaccional o no. Si es transaccional, puede gestionar sus propias transacciones del Servicio de transacción de objetos (OTS) o utilizar transacciones del OTS gestionadas por contenedor. Para obtener una explicación de la relación entre transacciones del OTS y unidades de trabajo de CICS, consulte "Enterprise beans: gestión de transacciones" en la página 241.
- No es recuperable: si el servidor EJB se bloquea, puede resultar destruido.
- Puede ser de dos tipos: **con estado** y **sin estado**.

Beans de sesión con estado:

Un bean de sesión con estado tiene un estado conversacional *específico del cliente*, que mantiene en distintos métodos y transacciones; por ejemplo, un objeto de "carro de la compra" mantendría una lista de los elementos seleccionados para comprar por el usuario.

Un bean de sesión con estado que gestiona sus propias transacciones puede comenzar una transacción del OTS en un método y confirmarla o retrotaerla en un método posterior.

Beans de sesión sin estado:

Un bean de sesión sin estado no tiene un estado no transitorio específico de cliente (ni de ningún otro tipo); por ejemplo, un objeto "cotización en bolsa" puede devolver los precios actuales de las acciones.

Un bean de sesión sin estado que gestione sus propias transacciones y que comience una transacción debe confirmar (o retrotraer) la transacción en el mismo método en el que la haya iniciado.

Beans de entidad:

CICS no soporta los beans de entidad directamente. Es decir, los beans de entidad no se pueden ejecutar en un servidor EJB de CICS. Sin embargo, un bean de sesión o un programa que se ejecute en un servidor EJB de CICS puede ser un cliente de un bean de entidad que se ejecute en un servidor EJB que no sea de CICS.

Importante

Un bean de entidad:

- Generalmente es una representación de objeto de datos empresariales, como un pedido de cliente. Normalmente, los datos:
  - Se mantienen en un almacén de datos permanente, como una base de datos.
  - Tienen que persistir más allá de la duración de una instancia de cliente. Por lo tanto, un bean de entidad tiene una duración relativamente larga en comparación con un bean de sesión.
- Al objeto puede acceder más de un cliente a la vez. Esto es posible debido a que cada instancia de un bean de entidad se identifica mediante una **clave primaria**, que se puede utilizar para encontrarlo mediante la interfaz inicial.
- Puede gestionar su propia persistencia (**persistencia gestionada por bean**) o delegar la tarea en su contenedor (**persistencia gestionada por contenedor**).

Si el bean gestiona su propia persistencia, el desarrollador del bean debe implementar operaciones de persistencia —por ejemplo, llamadas a JDBC o SQL— directamente en el bean.

Si el bean de entidad delega la persistencia en el contenedor, este gestiona el estado persistente de forma transparente; el desarrollador del bean no tiene que codificar ninguna operación de persistencia en el bean.

- Puede ser transaccional o no. Si es transaccional, todas las funciones de transacción las realizan implícitamente el contenedor EJB y el servidor. No hay sentencias de demarcación de transacción en el código del bean. A diferencia de los beans de sesión, un bean de sesión no tiene permiso para gestionar sus propias transacciones del OTS. Consulte "Enterprise beans: gestión de transacciones" en la página 241.
- Es recuperable: sobrevive al bloqueo de un servidor.

Comparación entre beans de sesión y beans de entidad:

Este es un resumen de las diferencias entre los beans de entidad y los beans de sesión.

Tabla 15. Comparación de beans de sesión y beans de entidad

Bean de sesión	Bean de entidad
Representa una única conversación con un cliente. Generalmente, incluye una o varias acciones que realizar en datos empresariales.	Normalmente incluye datos empresariales persistentes; por ejemplo, una fila en una base de datos.
Tiene una duración relativamente corta.	Tiene una duración relativamente larga.
Lo crea y lo utiliza un único cliente.	Pueden compartirlo varios clientes.
No tiene clave primaria.	Tiene una clave primaria que permite que más de un cliente encuentre una instancia y la comparta.
Generalmente, persiste solo durante la vida de la conversación con el cliente. (Sin embargo, puede optar por guardar información).	Persiste más allá de la duración de una instancia de cliente. La persistencia puede estar gestionado por contenedor o por bean.
No es recuperable: si el servidor EJB sufre una anomalía, puede resultar destruido.	Es recuperable: sobrevive a las anomalías del servidor EJB.
Puede ser con estado (es decir, tener un estado específico del cliente) o sin estado (no tiene ningún estado persistente).	Generalmente es con estado.

Tabla 15. Comparación de beans de sesión y beans de entidad (continuación)

Bean de sesión	Bean de entidad
Puede ser transaccional o no. Si es transaccional, puede gestionar sus propias transacciones del OTS o utilizar transacciones gestionadas por contenedor. Un bean de sesión con estado que gestione sus propias transacciones puede comenzar una transacción del OTS en un método y confirmarla o retrotaerla en un método posterior. Un bean de sesión sin estado que gestione sus propias transacciones y que comience una transacción del OTS debe confirmar (o retrotraer) la transacción en el mismo método en el que fuera iniciada. El estado de un bean de sesión con estado y transaccional no se retrotrae automáticamente en una retrotracción de transacción. En algunos casos, el bean puede utilizar sincronización de sesión para reaccionar ante un punto de sincronización.	Puede ser transaccional o no. Debe utilizar el modelo de transacción gestionada por contenedor. Si es transaccional, su estado se retrotrae automáticamente en una retrotracción de transacción.
No es reentrante.	Puede ser reentrante.

Enterprise beans: gestión de transacciones

Los clientes pueden comenzar, confirmar y retrotraer transacciones ACID utilizando una implementación del Java Transaction Service (JTS) o del CORBA Object Transaction Service (OTS).

Estas transacciones ACID [1]son análogas a las unidades de trabajo distribuidas de CICS. En este manual empleamos el término **transacción del OTS** para diferenciar estas transacciones de las definiciones de transacción de CICS (las que cuentan con identificadores de transacción de 4 caracteres) y de las instancias de transacción de CICS (que a veces se llaman, de forma poco precisa, “tareas”).

Cuando un cliente llama a un enterprise bean en el ámbito de una transacción del OTS, la información sobre dicha transacción fluye al servidor EJB en un “contexto de servicio” de IIOP, que es como un parámetro extra (oculto) en la solicitud de método. El servidor EJB utiliza esta información si tiene que participar en la transacción. Si el método de un enterprise bean tiene que ejecutarse en la transacción del OTS de un cliente (si hay alguna) se determina mediante el valor del **atributo de transacción** especificado en el descriptor de despliegue del bean. El método puede ejecutarse en la transacción del OTS del cliente, en una transacción del OTS separada que se cree para la duración del método o en una transacción que no sea del OTS.

Los beans de entidad deben utilizar **transacciones del OTS gestionadas por contenedor**. Todas las funciones de transacción las realizan implícitamente el contenedor EJB y el servidor. No hay sentencias de demarcación de transacción en el código del bean.

1. Transacciones que poseen atomicidad, consistencia, aislamiento y durabilidad. Jim Gray and Andreas Reuter, *Transaction Processing: Concepts and Techniques*, 1993.

Los beans de sesión pueden utilizar transacciones del OTS gestionadas por contenedor o **transacciones del OTS gestionadas por bean**. Un bean de sesión que utilice transacciones gestionadas por bean emplea métodos de la interfaz javax.transaction.UserTransaction para la demarcación de las transacciones. Un bean de sesión con estado que gestione sus propias transacciones puede comenzar una transacción del OTS en un método y confirmarla o retrotaerla en un método posterior. Un bean de sesión sin estado que gestione sus propias transacciones y que comience una transacción del OTS debe confirmar (o retrotraer) la transacción en el mismo método.

En el tiempo de ejecución, el contenedor EJB implementa servicios de transacción en función del valor del atributo de transacción especificado en el descriptor de despliegue del bean. Los valores posibles del atributo de transacción son:

Mandatory
Indica que el bean se debe ejecutar siempre en el contexto de la transacción del OTS del interlocutor. Si el interlocutor no cuenta con ninguna transacción cuando llama al bean, el contenedor genera una excepción javax.transaction.TransactionRequiredException y se produce un error en la solicitud.

Never
Indica que el bean no se debe invocar en el contexto de una transacción del OTS. Si un interlocutor tiene una transacción del OTS cuando llama al bean, el contenedor genera una excepción java.rmi.RemoteException y se produce un error en la solicitud.

NotSupported
Indica que el bean no se puede ejecutar en el contexto de una transacción del OTS. Si un interlocutor tiene una transacción del OTS cuando llama al bean, el contenedor suspende la transacción durante la duración de la llamada a método y reanuda la transacción suspendida cuando el método se ha completado. El contexto de la transacción suspendida del cliente no se pasa a los gestores de recursos ni a los objetos de enterprise bean que se invoquen desde el método.

Required
Indica que el bean se debe ejecutar en el contexto de una transacción del OTS. Si un interlocutor tiene una transacción del OTS cuando llama al bean, el método participa en la transacción del interlocutor. Si el interlocutor no cuenta con ninguna transacción del OTS, el contenedor inicia una nueva transacción del OTS para el método.

RequiresNew
Indica que el bean se debe ejecutar en el contexto de una nueva transacción del OTS. El contenedor siempre inicia una nueva transacción del OTS para el método. Si el interlocutor tiene una transacción del OTS cuando llama al bean, el contenedor suspende la transacción durante la duración de la llamada a método. El contexto de la transacción suspendida del cliente no se pasa a los gestores de recursos ni a los objetos de enterprise bean que se invoquen desde el método.

Supports
Indica que el bean se puede ejecutar con o sin un contexto de transacción. Si un interlocutor tiene una transacción del OTS cuando llama al bean, el método participa en la transacción del interlocutor. Si el interlocutor no cuenta con ninguna transacción del OTS, el método se ejecuta sin ninguna.

Nota: Los métodos de enterprise bean siempre se ejecutan en una tarea de CICS, bajo una unidad de trabajo de CICS. Incluso si un método de enterprise bean no se ejecuta en ninguna transacción del OTS, cualquier actualización que el método realiza en recursos recuperables se confirma únicamente con la terminación normal de la tarea de CICS y se deshacen los cambios si hay necesidad de una retrotracción.

El valor del atributo de transacción de un método determina si la tarea de CICS en la que se ejecuta el método hace que su unidad de trabajo forme parte de una transacción del OTS distribuida más amplia o no lo hace.

Una única tarea de CICS no puede contener más de un enterprise bean, ya que CICS trata una ejecución de un enterprise bean como el inicio de una nueva tarea. Puede crear una aplicación que incluya más de un enterprise bean, pero dicha aplicación no funcionará como una tarea única de CICS.

Visión general de la seguridad de Enterprise Beans

La seguridad de EJB tiene relación con la autenticación, el control de accesos y el mecanismo de la política de seguridad de Java 2.

Autenticación:

La autenticación de clientes EJB utiliza el protocolo de capa de sockets seguros (SSL) de TCP/IP.

Consulte Support for security protocols, en la *Guía de seguridad RACF de CICS*, para obtener información sobre la configuración de CICS para utilizar SSL.

Control de accesos:

El acceso a métodos de enterprise bean se basa en el concepto de roles de seguridad. Puede utilizar la seguridad de transacción y la seguridad de recursos de CICS con los recursos de EJB.

Roles de seguridad:

El acceso a métodos de enterprise bean se basa en el concepto de **roles de seguridad**. Un rol de seguridad representa un tipo de usuario de una aplicación en términos de los permisos que dicho usuario debe tener para utilizar correctamente esta aplicación.

Los roles que tienen permiso para ejecutar un enterprise bean concreto o método concretos de un bean se especifican en el descriptor de despliegue del bean, y la correlación de roles de seguridad con usuarios individuales se realiza en el gestor de seguridad externa.

Para obtener más información sobre roles de seguridad, consulte “Roles de seguridad” en la página 367.

Seguridad de transacción y seguridad de recursos de CICS:

Puede utilizar la seguridad de transacción y la seguridad de recursos de CICS con los recursos de EJB.

La seguridad de transacción de CICS se aplica a las transacciones de CICS asociadas con métodos de enterprise bean: es decir, las transacciones indicadas en las definiciones REQUESTMODEL de EJB.

La seguridad de recursos de CICS se aplica a los recursos de CICS a los que acceden los enterprise beans (por ejemplo, mediante JCICS).

El gestor de seguridad de Java:

La seguridad del entorno del contenedor de enterprise beans está protegida por el mecanismo de política de seguridad de Java y es independiente de la seguridad de CICS. El mecanismo de política de seguridad es uno de los componentes que conforman el modelo de seguridad Java.

El mecanismo de política de seguridad se utiliza para imponer las restricciones en la especificación de EJB referentes a las funciones de Java que no pueden emitir los enterprise beans. CICS proporciona un archivo de política que impone este comportamiento.

Para utilizar JDBC o SQLJ desde enterprise beans con un mecanismo de política de seguridad Java activo, debe utilizar el controlador de JDBC 2.0 que proporciona DB2. El controlador del servidor de datos de IBM para JDBC y SQLJ proporcionado por DB2 no admite la seguridad de Java y sufrirá una anomalía con una excepción de seguridad a no ser que se inhabilite el mecanismo.

Enterprise beans: tareas de usuario

Los roles implicados en el desarrollo y el despliegue de aplicaciones que utilizan enterprise beans son: proveedor de bean, ensamblador de aplicaciones, desplegador y administrador del sistema.

Nota: En organizaciones de menor tamaño, una sola persona puede ser responsable de más de uno de estos roles.

El proveedor de bean:

El proveedor de bean desarrolla enterprise beans reutilizables que generalmente implementan tareas de negocio o entidades empresariales.

La salida del proveedor de beans es un **archivo JAR EJB** que contiene uno o más enterprise beans. El proveedor de bean es responsable de lo siguiente:

- Las clases Java que implementan los métodos de negocio de un enterprise bean.
- La definición de las interfaces inicial y de componente del bean.
- El descriptor de despliegue del bean.

  El descriptor de despliegue incluye la información sobre la estructura —por ejemplo, el nombre de la clase del enterprise bean— del enterprise bean y declara todas las dependencias externas del bean, como, por ejemplo, los nombres y los tipos de los gestores de recursos que utiliza el enterprise bean.

El ensamblador de aplicaciones:

El ensamblador de aplicaciones crea aplicaciones que utilizan enterprise beans. Combina enterprise beans y código de cliente escrito manualmente en una aplicación cliente/servidor. Aunque debe estar familiarizado con la funcionalidad

que brindan el componente de enterprise beans y las interfaces iniciales, no es necesario que tenga ningún conocimiento sobre la implementación de enterprise beans.

La entrada para el ensamblador de aplicaciones la componen uno o más archivos JAR EJB producidos por el proveedor del bean. Su salida consta de uno o más archivos JAR EJB que contienen los enterprise beans, junto con sus instrucciones de ensamblaje de aplicaciones y los valores de entorno personalizados. Ha insertado las instrucciones de ensamblaje de aplicaciones, los roles de seguridad y los valores de entorno en los descriptores de despliegue.

El ensamblador de aplicaciones también puede combinar enterprise beans con otros tipos de componentes de aplicación —por ejemplo, JavaBeans— al ensamblar una aplicación.

Generalmente, el paso de ensamblaje de aplicaciones se produce antes del despliegue de los enterprise beans. Sin embargo, en algunas ocasiones el ensamblaje se puede realizar tras el despliegue de todos los enterprise beans o de parte de ellos.

El desplegador:

El desplegador toma uno o más archivos JAR EJB producidos por el ensamblador de aplicaciones y despliega los enterprise beans contenidos en los archivos JAR EJB en un CorbaServer específico de un servidor EJB.

El desplegador debe:

- Resolver todas las dependencias externas declaradas por el proveedor de bean. Por ejemplo, debe asegurarse de que todas las fábricas de conexiones del gestor de recursos utilizadas por los enterprise beans están presentes en el entorno operativo y de enlazarlos con las referencias de la fábrica de conexiones del Gestor de recursos declaradas en el descriptor de despliegue.
- Siga las instrucciones de ensamblaje de aplicaciones definidas por el ensamblador de aplicaciones. Por ejemplo, el desplegador es responsable de la correlación de los roles de seguridad definidos por el ensamblador de aplicaciones para grupos de usuarios y perfiles de gestor de seguridad externo de CICS.

El proceso de despliegue es semiautomático. Para realizar este rol, el desplegador utiliza una **herramienta de despliegue**. Las herramientas de despliegue las proporciona CICS.

La salida del desplegador son enterprise beans que se han personalizado para el entorno operativo de destino y que se han desplegado en uno o más CorbaServer.

El administrador del sistema:

El administrador del sistema es responsable de configurar y administrar las regiones CICS que componen el servidor EJB lógico junto con las conexiones de red. También es responsable de supervisar el bienestar de las aplicaciones EJB desplegadas en el tiempo de ejecución.

Visión general del despliegue de enterprise beans

Un bean Java de escritorio se desarrolla, se instala y se ejecuta en una estación de trabajo. Un enterprise bean que se ejecute en un servidor necesita una fase adicional, el **despliegue**, para preparar el bean para el entorno de ejecución e instalarlo en el servidor EJB.

Los enterprise beans los produce el proveedor del bean y los personaliza el ensamblador de aplicaciones. El ensamblador de aplicaciones puede utilizar una herramienta como el Assembly Toolkit (ATK) (descrito en The enterprise bean deployment tool, ATK, en la *Guía de operaciones y programas de utilidad de CICS*) para personalizar el archivo JAR EJB. El archivo JAR EJB que se pasa al desplegador contiene:

- Las clases Java para uno o más enterprise beans.
- Un único descriptor de despliegue, escrito en XML, que describe las características de cada uno de los enterprise beans, como:
  - Atributos de transacción
  - Propiedades del entorno
  - Niveles de seguridad
  - Información de ensamblaje de aplicaciones.

También se necesita información específica para CICS, como requisitos de definición de recurso.

A continuación tiene un compendio del proceso de despliegue:

1. Una herramienta de despliegue, como la herramienta de despliegue de enterprise bean, ATK. Utilice esta herramienta para transformar el archivo JAR EJB en un archivo JAR deplegable, adecuado para el despliegue. El archivo transformado contiene el descriptor de despliegue XML y las clases de enterprise bean del archivo JAR EJB, más clases adicionales generadas para el soporte del contenedor EJB. El archivo transformado se almacena como un archivo JAR desplegado en el sistema de archivos z/OS UNIX.

   Almacene el archivo JAR desplegado que se almacena en el directorio del archivo JAR desplegado del CorbaServer (especificado por la opción DJARDIR de la definición CORBASERVER). El directorio del archivo JAR desplegado también se conoce como **directorio "de recogida"**. Cuando CICS explora el directorio de recogida, automáticamente crea e instala una definición de cada archivo JAR desplegado nuevo o actualizado que encuentra ahí. CICS explora el directorio de recogida en cualquiera de las situaciones siguientes:

   - Automáticamente, cuando se instala la definición CORBASERVER.
   - Cuando se indica por medio de un mandato **EXEC CICS** o **CEMT PERFORM CORBASERVER SCAN** explícito.
   - Cuando se indica por medio del gestor de recursos de los enterprise beans (también conocido como el RM para enterprise beans), que emite un mandato **PERFORM CORBASERVER SCAN** en su nombre. (El gestor de recursos para los enterprise beans se describe en The Resource Manager for Enterprise Beans, en la *Guía de operaciones y programas de utilidad de CICS*.

2. Las definiciones de recursos de CICS son necesarias para:
   - El entorno de ejecución del CorbaServer (CORBASERVER). (La misma definición CORBASERVER se instalará en cada AOR CICS del servidor EJB lógico).
   - Servicios TCP/IP (para IIOP). Una o más definiciones TCPIPSERVICE se instalarán en cada región CICS del servidor EJB lógico.

- Modelos de solicitud, para asociar solicitudes IIOP de cliente con TRANSID de CICS (y así asociar métodos de bean con conjuntos de características de ejecución, que abarcan aspectos como la seguridad, la prioridad y la supervisión). Los modelos de solicitud solo son necesarios si el TRANSID predeterminado, CIRP, no es adecuado. (Tal vez desee segregar la carga de trabajo de IIOP por ID de transacción, por ejemplo).

  Nota: Puede utilizar la transacción CREA provista por CICS para visualizar los ID de transacción asociados con beans y métodos de bean concretos en el CorbaServer. Puede cambiar los ID de transacción, aplicar los cambios y guardar dichos cambios en las nuevas definiciones REQUESTMODEL.
- Archivos JAR desplegados (DJAR), cada uno de los cuales incluye el nombre de archivo z/OS UNIX de un archivo JAR desplegado. Si almacena los archivos JAR desplegados en el directorio de "recogida" del CorbaServer, las definiciones DJAR se crean y se instalan automáticamente cuando se instala el CorbaServer (o cuando se produce una exploración posterior).

Nota: "Configuración de un servidor EJB lógico" en la página 251 contiene más información sobre estas definiciones de recursos en línea.

3. Las definiciones de seguridad se añaden al gestor de seguridad externa. Estas especifican qué roles pueden ejecutar beans y métodos concretos y qué ID de usuario se asocian con cada rol.
4. Las definiciones de recurso se instalan en CICS. Instalar una definición DJAR hace que CICS:
   - Copie el archivo JAR desplegado (y las clases que contiene) en un directorio de "estantería" de z/OS UNIX. El **directorio de estantería** es donde CICS mantiene copias de los archivos JAR desplegados.
   - Lea el JAR desplegado desde la estantería, analice su descriptor de despliegue XML y almacene la información que contiene.

   Nota: Si almacena los archivos JAR desplegados en el directorio de "recogida" del CorbaServer, las definiciones de DJAR se instalan automáticamente cuando se instala el CorbaServer (o cuando se produce una exploración posterior).
5. Se publica una referencia a la clase de interfaz inicial de cada bean desplegado en un espacio de nombres externo. Los clientes pueden acceder al espacio de nombres mediante JNDI.

   Si se especifica AUTOPUBLISH(YES) en la definición de CORBASERVER, el contenido de un archivo JAR desplegado se publica automáticamente en el espacio de nombres cuando la definición DJAR se instala correctamente en el CorbaServer. Como alternativa, puede emitir un mandato **PERFORM CORBASERVER PUBLISH** o **PERFORM DJAR PUBLISH**.

La Figura 14 en la página 248 muestra el proceso de despliegue.

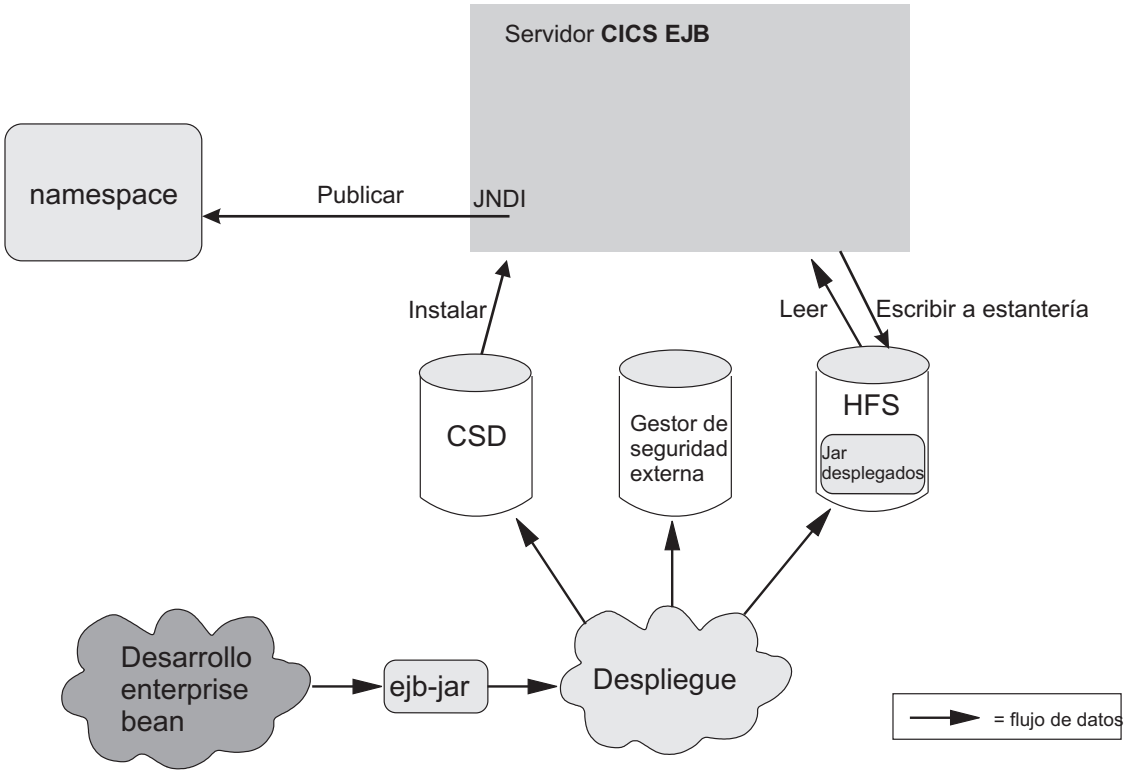


Figura 14. Despliegue de enterprise beans en un servidor EJB de CICS. Se utiliza una herramienta de despliegue para realizar la generación de código en el archivo JAR EJB que contiene las clases de bean. El archivo transformado se almacena como un archivo JAR desplegado en el sistema de archivos z/OS UNIX. Se crea una definición de recursos en línea del archivo JAR desplegado y se instala en CICS, junto con otras definiciones para servicios TCP/IP, modelos de solicitud y el entorno de ejecución de CorbaServer. Las definiciones de seguridad se crean en el gestor de seguridad externa.

Visión general de la configuración de CICS como un servidor EJB

Un servidor EJB de CICS contiene estos componentes básicos.

La región de escucha
El trabajo de la región de escucha es escuchar las solicitudes de conexión TCP/IP entrantes (y responder a ellas). Un recurso TCPIPSERVICE configura una región de escucha IIOP para escuchar a un puerto TCP/IP determinado y para adjuntar un **receptor de peticiones** IIOP con el fin de manejar cada conexión.

Cuando se ha establecido una conexión IIOP entre un programa cliente y un receptor de peticiones determinado, todas las solicitudes posteriores del programa cliente a través de esa conexión se dirigen al mismo receptor de peticiones.

El receptor de peticiones
El receptor de peticiones analiza los datos de IIOP estructurados. Pasa la solicitud entrante a un **procesador de solicitudes** mediante una **secuencia de solicitudes**, que es un mecanismo de direccionamiento interno de CICS. La clave de objeto de la solicitud determina si la solicitud se debe enviar a un procesador de solicitudes nuevo o a uno ya existente.

Si la solicitud se debe enviar a un nuevo procesador de solicitudes, se determina un identificador de transacción de CICS comparando los datos de

solicitud con plantillas definidas en recursos REQUESTMODEL. (Si no se puede encontrar ningún recurso REQUESTMODEL coincidente, se utiliza la transacción predeterminada, CIRP). El TRANSID define parámetros de ejecución que utiliza el procesador de solicitudes.

El procesador de solicitudes
El procesador de solicitudes es una instancia de transacciones que gestiona la ejecución de la solicitud de IIOP. Este:

- Localiza el objeto identificado por la solicitud.
- Para una solicitud de enterprise bean, llama al contenedor para que procese el método de bean.
- Para una solicitud para un objeto CORBA sin estado, generalmente el ORB procesa la propia solicitud (aunque el servicio de transacción puede estar implicado también).

Para obtener información completa sobre regiones de escucha, receptores de peticiones y procesadores de solicitudes, consulte “El flujo de solicitudes de IIOP” en la página 386.

La Figura 15 en la página 250 muestra un servidor lógico de CICS. En este ejemplo, las regiones de escucha y las AOR están en grupos distintos, se utiliza optimización de la conexión para equilibrar las conexiones de cliente en las regiones de escucha y se emplea direccionamiento distribuido para equilibrar las transacciones del OTS en las AOR.

El servidor lógico consta de un conjunto de regiones de escucha clonadas y de un conjunto de AOR clonadas. En este ejemplo se utiliza optimización de la conexión mediante registro de DNS dinámico para equilibrar las conexiones de cliente en las regiones de escucha. El direccionamiento distribuido se utiliza para equilibrar transacciones del OTS en las AOR.

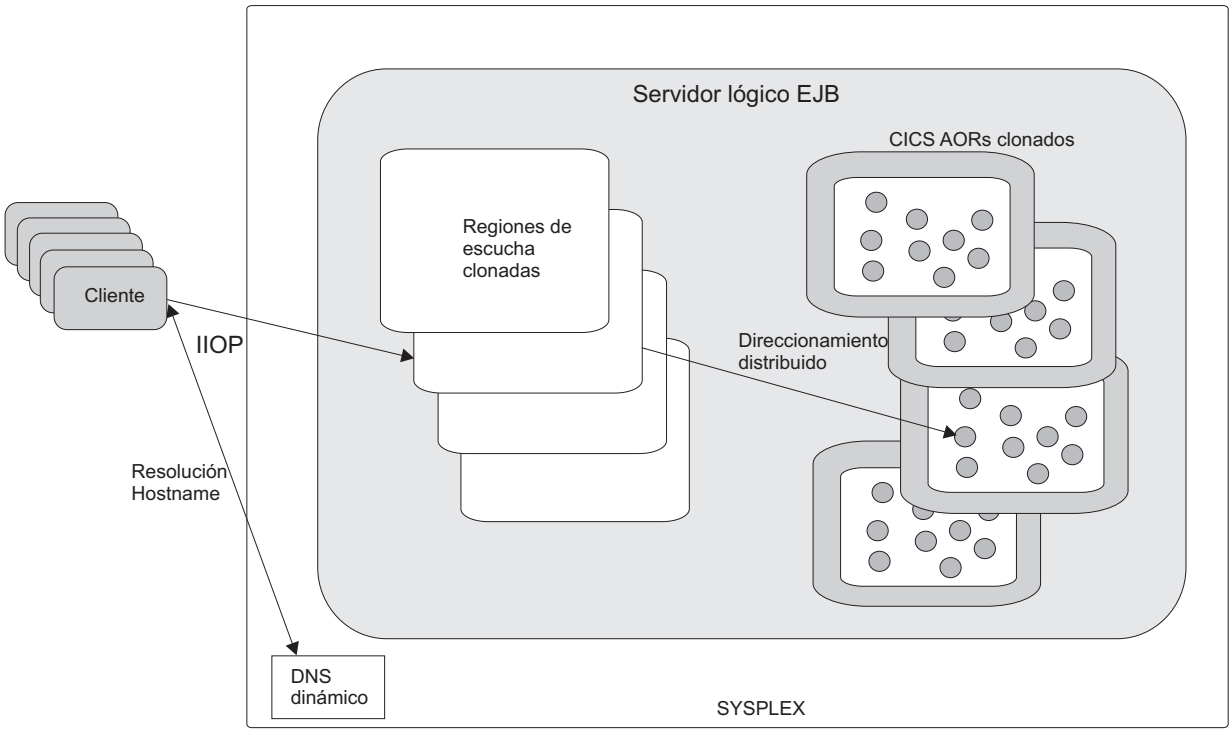


Figura 15. Un servidor EJB lógico de CICS

Servidores lógicos: enterprise beans en un sysplex:

Puede implementar un servidor EJB de CICS en una única región CICS.

Sin embargo, en un sysplex es probable que desee crear un servidor que conste de varias regiones. El uso de varias regiones hace que el error de una única región no sea tan crítico y permite utilizar el direccionamiento de carga de trabajo. Un servidor EJB lógico de CICS consta de una o más regiones CICS configuradas para comportarse como un único servidor EJB.

Por lo general, un servidor EJB lógico de CICS consta de los siguientes componentes:

- Un conjunto de regiones de escucha clonadas definidas por definiciones TCPIPSERVICE idénticas para escuchar las solicitudes de IIOP entrantes.
- Un conjunto de regiones propietarias de la aplicación (AOR) clonadas, cada una de las cuales admite un conjunto idéntico de clases de enterprise bean en un servidor CORBA definido de forma idéntica.

Nota: Las regiones de escucha y las AOR pueden separarse o combinarse en AOR de escucha.

Direccionamiento de la carga de trabajo en un sysplex:

El direccionamiento de carga de trabajo se implementa en dos niveles al dirigir las conexiones de cliente en las regiones de escucha y direccionar las transacciones del OTS en las AOR.

1. Para direccionar las conexiones de cliente en las regiones de escucha, puede utilizar cualquiera de los siguientes métodos:

- Optimización de conexiones mediante registro dinámico del Sistema de nombres de dominio (DNS).
- Direccionamiento de IP.
- Una combinación de optimización de conexiones y direccionamiento de IP.

Con la optimización de conexiones mediante registro dinámico de DNS, por ejemplo, se inician varias regiones CICS para que escuchen solicitudes de IIOP en el mismo puerto (mediante direcciones IP virtuales). Cada solicitud de conexión IIOP de cliente contiene un nombre de host genérico y un número de puerto. El nombre de host genérico de cada solicitud de conexión se resuelve en una dirección IP real mediante los servicios de MVS DNS y gestión de carga de trabajo (WLM).

2. Para direccionar las transacciones del OTS en las AOR, puede utilizar cualquiera de los siguientes métodos:
   - CICSPlex SM
   - Una versión personalizada del programa de direccionamiento distribuido de CICS, DFHDSRP.

   Importante

   Cuando se utiliza el programa de direccionamiento distribuido, resulta pertinente hablar sobre el direccionamiento dinámico de transacciones del OTS en las AOR. Sin embargo, específicamente lo que se direcciona de forma dinámica son las *solicitudes de método* para enterprise beans y los objetos sin estado CORBA. Existe un correspondencia entre el direccionamiento dinámico de las solicitudes de métodos y el direccionamiento dinámico de las transacciones del OTS: CICS invoca al programa de direccionamiento para solicitudes de métodos que se ejecutarán en una *nueva* transacción del OTS, pero no para solicitudes de métodos que se ejecutarán en una transacción del OTS *existente*; estas las dirige automáticamente a la AOR en la que se ejecuta la transacción del OTS existente. Sin embargo, debido a que las solicitudes de métodos que se ejecutarán en *una transacción que no sea del OTS* también se pueden direccionar de forma dinámica, la correspondencia no es exacta.

   Es importante comprender la diferencia entre las transacciones OTS nuevas y existentes.

   a. Una *nueva* transacción de OTS es una transacción en la que ya no participa el servidor lógico de destino, antes de la llamada al método actual; *no* necesariamente una transacción de OTS que se ha iniciado inmediatamente antes de la llamada al método.
   b. Una transacción de OTS *existente* es una transacción en la que el servidor lógico de destino ya participa, antes de la llamada al método actual; *no* una transacción de OTS que se ha iniciado hace algún tiempo.

   Por ejemplo, si un cliente inicia una transacción del OTS, realiza algún trabajo y luego llama a un método en un enterprise bean, en lo que respecta al servidor EJB de CICS, se trata de una nueva transacción del OTS, ya que no se ha llamado antes al servidor dentro del ámbito de esta transacción. Si luego el cliente realiza una segunda y una tercera llamada a método para el mismo objeto de destino, antes de confirmar su transacción de OTS, estas llamadas segunda y tercera se producen dentro del ámbito de la transacción del OTS existente.

Configuración de un servidor EJB lógico:

Debe realizar varios pasos para configurar un servidor EJB de CICS lógico para el soporte de enterprise beans.

Antes de configurar un servidor EJB lógico, asegúrese de que las regiones de dicho servidor EJB lógico, tanto de escucha como AOR, estén en el mismo nivel de CICS.

Siga estos pasos para configurar un servidor EJB de CICS lógico para el soporte de enterprise beans:

1. Cree un conjunto de regiones de escucha CICS Transaction Server para z/OS, Versión 4 Release 2 clonadas. Cada región de escucha debe tener el parámetro de inicialización del sistema **IIOPLISTENER** establecido en YES.
2. Cree un conjunto de AOR CICS Transaction Server para z/OS, Versión 4 Release 2 clonadas. Cada AOR debe cumplir estos criterios:
   - Estar configurada para utilizar JNDI.
   - Utilizar el mismo contexto inicial de JNDI que el resto de AOR.
   - Estar conectada a todas las regiones de escucha mediante MRO (no mediante ISC).
   - Estar configurada con el valor del parámetro de inicialización del sistema **IIOPLISTENER** definido como NO.
3. Cree un directorio raíz de estantería en z/OS UNIX. Por ejemplo, puede crear un directorio llamado `/var/cicsts/`. Para hacerlo, necesita un ID de usuario de z/OS UNIX con autorización de grabación en la vía de acceso del directorio que va a utilizar CICS. Tras haber creado el directorio de estantería, debe conceder a los ID de usuario de la AOR acceso completo de lectura, grabación y ejecución al directorio.
4. Cree un directorio (de recogida) del archivo JAR desplegado en z/OS UNIX. Por ejemplo, puede crear un directorio llamado `/var/cicsts/pickup/`. Las AOR deben tener, como mínimo, acceso de lectura a este.

   Si las AOR van a contener más de un entorno de ejecución de CorbaServer:
   - Debe crear un directorio de recogida distinto para cada CorbaServer.
   - Asigne distintos conjuntos de ID de transacción a los objetos soportados por cada CorbaServer. Es decir, cada CorbaServer de una AOR soporta un conjunto distinto de ID de transacción. Para asignar ID de transacción a métodos de bean, utilice definiciones REQUESTMODEL; consulte el paso 5.
5. Cree las siguientes definiciones de recurso. Puede crearlas en un CSD que compartan todas las regiones del servidor lógico, copiarlas todas a los CSD que utilizan las regiones o añadirlas a una descripción de recursos de CICSPlex SM que se aplique a todas las regiones. Como opción, puede utilizar el mecanismo de exploración de CICS, el Gestor de recursos para enterprise beans y la transacción proporcionada por CICS, CREA, para crear algunas de estas definiciones, como se describe a continuación.
   - Una TCPIPSERVICE.
     - En la opción PROTOCOL, especifique IIOP.
     - En la opción SSL, especifique NO.
     - En la opción AUTHENTICATE, especifique NO. Con esta especificación, el servicio de este puerto acepta solicitudes IIOP de entrada sin autenticar.
   - Algunas definiciones de REQUESTMODEL. En un servidor EJB de región única, las definiciones solo son necesarias si el TRANSID predeterminado, CIRP, no es adecuado. En un servidor lógico multirregión, sin embargo, las definiciones son necesarias si se desean dirigir solicitudes de método en varias AOR. La definición TRANSACTION para CIRP especifica DYNAMIC(NO). Las definiciones también son necesarias si, por ejemplo desea segregar la carga de trabajo de IIOP por ID de transacción.

Nota:

a. El atributo BEANNAME de cada definición REQUESTMODEL debe "coincidir" (en el sentido de coincidencia de patrón) con el nombre de un enterprise bean en el descriptor de despliegue de un archivo JAR desplegado en z/OS UNIX. El valor del atributo CORBASERVER debe coincidir, de forma literal o en el sentido de coincidencia de patrón, con el CorbaServer de la definición CORBASERVER.

b. Copie la definición de transacción para el TRANSID indicado en REQUESTMODEL de esa CIRP. Defina el atributo DYNAMIC como YES. Puede modificar cualquiera de los demás atributos, pero el nombre de programa debe ser el de un programa de JVM con una acJVMClass de com.ibm.cics.iiop.RequestProcessor.

c. Cuando el CorbaServer esté operativo, puede utilizar la transacción CREA provista por CICS para visualizar los ID de transacción asociados con beans y métodos de bean concretos en el CorbaServer. Puede cambiar los ID de transacción, aplicar los cambios y guardar dichos cambios en las nuevas definiciones REQUESTMODEL.

- Una definición CORBASERVER.

  El valor de la opción HOST de la definición CORBASERVER debe coincidir con la de la opción HOST o IPADDRESS de la definición TCPIPSERVICE. Sin embargo, si TCPIPSERVICE especifica un valor para DNSGROUP, la opción HOST de la definición CORBASERVER debe especificar un nombre de host genérico coincidente.

  En la opción UNAUTH, especifique el nombre de la definición TCPIPSERVICE. Debe especificar siempre un valor para el atributo UNAUTH cuando defina un CorbaServer, aunque pretenda que se autentiquen todas las solicitudes de entrada a este CorbaServer. Este valor es necesario porque el número de puerto de TCPIPSERVICE se utiliza para construir Interoperable Object References (IOR) que se exporten desde este servidor lógico. Si especifica el nombre de otras definiciones TCPIPSERVICE en las opciones CLIENTCERT o SSLUNAUTH, puede hacer que las regiones de escucha escuchen otros puertos para distintos tipos de solicitudes IIOP de entrada autenticadas. Para obtener más información, consulte Recursos CORBASERVER en la Guía de definición de recursos y Recursos TCPIPSERVICE en la Guía de definición de recursos.

  En la opción SHELF, especifique el nombre completo del directorio de estantería de z/OS UNIX que creara en el paso 3. Como la definición CORBASERVER está instalada en todas las AOR en el servidor lógico, este directorio de estantería de "alto nivel" lo comparten todas ellas. Cada AOR crea automáticamente su propio subdirectorio bajo el directorio de estantería y un subdirectorio para el CorbaServer bajo este.

  En la opción DJARDIR option, especifique el nombre completo del directorio del archivo JAR desplegado de z/OS UNIX (directorio de recogida) que se crea en el paso 4. Como en el directorio de estantería, el directorio de recogida (o directorios, si los AOR contienen múltiplesCorbaServers) es compartido por todas los AOR en el servidor lógico. En cada AOR, cuando se instala una definición CORBASERVER, CICS explora el directorio de recogida del CorbaServer e instala cualquier archivo JAR desplegado que encuentre ahí. Los copia en su directorio de estantería y crea e instala de forma dinámica definiciones DJAR para ellos.

  Especifique AUTOPUBLISH(YES) para que CICS publique beans en el espacio de nombres de forma automática cuando se instale una definición DJAR de forma correcta.

En la opción STATUS, especifique Enabled.

- Definiciones de FILE para los siguientes archivos que CICS necesita:

  El directorio de EJB, DFHEJDIR
  Es un archivo que contiene un directorio de secuencias de solicitudes, que deben compartir todas las regiones, tanto de escucha como AOR, del servidor EJB lógico. Las secuencias de solicitudes se utilizan en el direccionamiento de solicitudes de método para enterprise beans y objetos sin estado CORBA. Debe definir DFHEJDIR como recuperable.

  El almacén de objetos EJB, DFHEJOS
  Es un archivo de beans de sesión con estado que se han desactivado. Deben compartirlo todas las AOR del servidor EJB lógico. Debe definirlo como no recuperable.

  Para compartir DFHEJDIR y DFHEJOS en varias regiones, puede, por ejemplo, utilizar cualquiera de los siguientes métodos:

  – Defínalos como archivos remotos en una región propietaria del archivo (FOR).
  – Defínalos como tablas de datos de recursos de acoplamiento.
  – Utilice VSAM RLS.

  Las definiciones de FILE de ejemplo se encuentran en los siguientes grupos:

  – En el caso de DFHEJDIR y DFHEJOS, se encuentran en el grupo de RDO proporcionado por CICS, DFHEJVS.
  – En el caso de DFHEJDIR y DFHEJOS, se encuentran en el grupo de RDO proporcionado por CICS, DFHEJCF.

  Las definiciones FILE de VSAM RLS para DFHEJDIR y DFHEJOS están en el grupo de RDO proporcionado por CICS, DFHEJVR. DFHEJVS, DFHEJCF y DFHEJVR no se incluyen en la lista de grupos de inicio de CICS predeterminada, DFHLIST.

Nota: Estos pasos asumen que el servidor lógico tiene un único CorbaServer. Para crear otro CorbaServer, cree una segunda definición CORBASERVER y otra definición TCPIPSERVICE.

6. Defina los conjuntos de datos VSAM subyacentes para DFHEJDIR y DFHEJOS. CICS proporciona JCL de ejemplo para ayudarle a crear este archivo en el miembro DFHDEFDS de la biblioteca SDFHINST.
7. Mediante una herramienta de despliegue como el Assembly Toolkit (ATK), tome uno o más archivos JAR EJB y realice en ellos generación de código para producir archivos JAR desplegados en z/OS UNIX. Almacene los archivos JAR desplegados en el directorio de recogida del CorbaServer.
8. Lance todas las regiones CICS. En cada región de escucha, las definiciones que instalar desde el CSD son las siguientes:
   - La definición TCPIPSERVICE.
   - Las definiciones REQUESTMODEL.
   - La definición de archivo para DFHEJDIR.

   En cada AOR, las definiciones que instalar desde el CSD son las siguientes:
   - La definición TCPIPSERVICE.
   - Las definiciones REQUESTMODEL.

     Las definiciones REQUESTMODEL de las AOR son necesarias para las solicitudes salientes a objetos locales. Si un objeto CORBA sin estado o un enterprise hace una llamada a otro objeto y dicho objeto está disponible en

la AOR local, CICS no envía la solicitud a una región de escucha. En su lugar, ejecuta el método llamado en la tarea actual ("bucle de retorno hermético") o inicia otro procesador de solicitudes en la AOR local ("bucle de retorno normal"). Si se utiliza el bucle de retorno normal, es preferible que la nueva tarea del procesador de solicitudes utilice el mismo REQUESTMODEL que el utilizado para llamar al primer objeto; en caso contrario pueden producirse resultados imprevisibles. Si los objetos CORBA sin estado y los enterprise beans no realizan llamadas salientes, los REQUESTMODEL de la AOR no son necesarios estrictamente.

- La definición CORBASERVER.
- Las definiciones de archivo para DFHEJDIR y DFHEJOS.

Si pone los archivos JAR desplegados en el directorio de recogida compartido del CorbaServer, se crean y se instalan automáticamente definiciones DJAR cuando se instala el CorbaServer o cuando se produce una exploración posterior. Cree definiciones DJAR estáticas (instaladas por CSD) solo para archivos JAR desplegados que se colocan en otros directorios de z/OS UNIX.

9. En cada AOR, cuando se instala la definición CORBASERVER, CICS explora el directorio de recogida e instala todos los archivos JAR desplegados que encuentra ahí. Los copia a su directorio de estantería y crea dinámicamente e instala definiciones DJAR para ellos.

   Es posible poner archivos JAR desplegados al instalar. Si lo hace, puede forzar a que CICS realice otra exploración emitiendo un mandato **CORBASERVER PERFORM SCAN**. Emita este mandato mediante **EXEC CICS**, la transacción del terminal maestro de CEMT o el Gestor de recursos basado en web para enterprise beans, conocido también como el RM para enterprise beans.

10. Como se ha especificado AUTOPUBLISH(YES) en la definición CORBASERVER, cuando las definiciones DJAR se instalan correctamente, los directorios de inicio de los enterprise beans se enlazan automáticamente con el espacio de nombres de JNDI.

    Si se especifica AUTOPUBLISH(NO), debe emitir un mandato **PERFORM CORBASERVER(nombre_CorbaServer) PUBLISH** en al menos una de las AOR. Debe emitir este mandato mediante **EXEC CICS**, la transacción del terminal maestro de CEMT, el RM para enterprise beans o la vista WUI de CICSPlex SM.

11. En el parámetro de inicialización del sistema DSRTPGM para las regiones de escucha, especifique el nombre del programa de direccionamiento distribuido que utilizar. Si utiliza CICSPlex SM, especifique el nombre del programa de direccionamiento de CICSPlex SM, EYU9XLOP. En caso contrario, especifique el nombre del programa de direccionamiento personalizado. Para obtener más información sobre el parámetro de inicialización del sistema DSRTPGM , consulte Parámetro de inicialización del sistema DSRTPGM en la Guía de definición de sistema.

La Figura 16 en la página 256 muestra las definiciones RDO necesarias para definir un servidor EJB lógico de CICS. Muestra qué definiciones son necesarias en las regiones de escucha, cuáles en las AOR y cuáles en ambas.

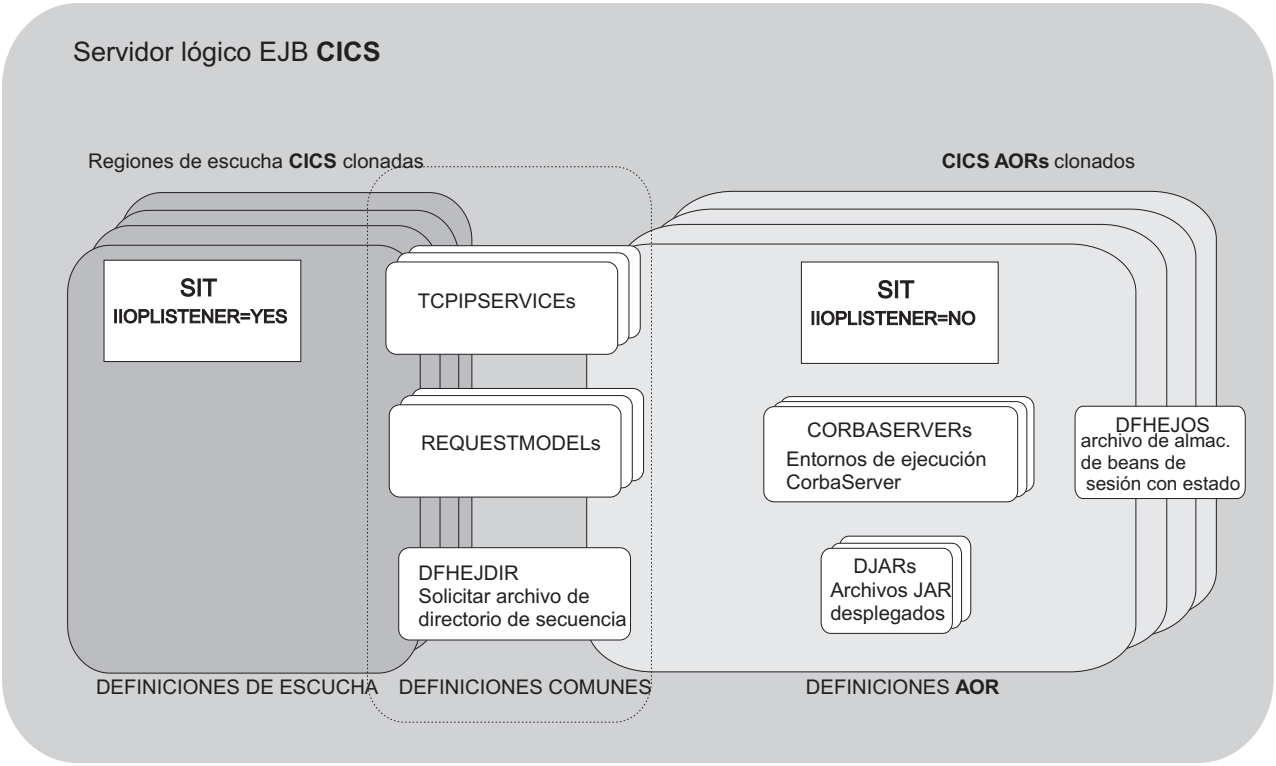


Figura 16. Definiciones de recurso en un servidor EJB lógico de CICS

Enterprise beans: ¿qué puede hacer un cliente con un bean?

Esta sección contiene fragmentos de código de ejemplo que ilustran de qué forma un programa cliente puede utilizar un enterprise bean.

Obtención de una referencia a la interfaz inicial del bean:

Para realizar cualquier cosa con el bean, el cliente debe obtener una referencia a la interfaz inicial de dicho bean.

Para ello, este busca un nombre conocido mediante la JNDI:

```
// Obtenga un contexto inicial de JNDI.
Context initContext = new InitialContext();

// Busque la interfaz inicial del bean.
Object accountBeanHome = initContext.lookup("prefijo_JNDI/AccountBean");
// donde:
// 'prefijo_JNDI/' es el prefijo de JNDI en la definición CORBASERVER.
// 'AccountBean' es el nombre del bean en el descriptor de despliegue XML.

// Convierta al tipo correcto.
AccountHome accountHome = (AccountHome)
            PortableRemoteObject.narrow(accountBeanHome,AccountHome.class);
```

Utilización de la interfaz inicial:

El cliente puede utilizar la interfaz inicial del bean para crear una nueva instancia del bean y suprimir una instancia del bean.

```
// Cree dos instancias del bean.
Account anAccount = accountHome.create();
Account anotherAccount = accountHome.create("12345");

// Elimine una instancia del bean.
accountHome.remove("12345");
```

Utilización de la interfaz de componente:

El cliente puede utilizar la interfaz de componente del bean para invocar los métodos del bean y suprimir dicho bean.

```
// Utilice el bean
anAccount.deposit(1000000);
// Elimínelo.
anAccount.remove();
```

Enterprise beans: ¿qué puede hacer un bean?

Un enterprise bean se beneficia de muchos servicios —como gestión de ciclo vital y seguridad— que el contenedor EJB proporciona implícitamente, en base a los valores que hay en el descriptor de despliegue.

Esto deja al proveedor del bean libre para concentrarse en la lógica empresarial de dicho bean. Esta sección analiza algunas de las cosas que puede hacer un bean.

Buscar entradas de JNDI
Un bean puede utilizar llamadas a JNDI para recuperar:

- Referencias a recursos.
- Variables de entorno.
- Referencias a otros beans.

Acceder a gestores de recursos
Un bean puede:

- Obtener una conexión con un gestor de recursos.
- Utilizar los recursos del gestor de recursos.
- Cerrar la conexión.

Enlazar con programas CICS.
Un bean puede utilizar JCICS o el CCI Connector for CICS TS para enlazar con un programa de CICS, que puede estar escrito en cualquiera de los lenguajes soportados por CICS y ser local o remoto. El proveedor del bean puede utilizar el CCI Connector for CICS TS para compilar enterprise beans que aprovechen la potencia de los programas CICS ya existentes (que no sean Java).

El CCI Connector for CICS TS se describe en “El CCI Connector for CICS TS” en la página 337.

Acceder a archivos
Un bean puede utilizar JCICS para leer archivos y grabar en ellos.

Llamar a otros beans
Un bean puede:

- Obtener referencias a las interfaces de componente e inicial de otros objetos de bean.
- Invocar los métodos de otro objeto de bean.
- Ser llamado desde otros objetos de bean.

Un bean puede actuar como cliente de otro objeto de bean, como servidor de otro objeto de bean o como ambos.

Tenga en cuenta que una única tarea de CICS (una instancia de una transacción) no puede contener más de un enterprise bean, ya que CICS trata una ejecución de un enterprise bean como el inicio de una nueva tarea. Puede crear una aplicación que incluya más de un enterprise bean, pero dicha aplicación no funcionará como una tarea única de CICS.

Gestionar transacciones
Como opción, un bean de sesión puede gestionar sus propias transacciones del OTS, en lugar de utilizar transacciones gestionadas por contenedor. Como alternativa, puede hacer que su transacción sea gestionada por el interlocutor.

Configuración de un servidor EJB

Este capítulo indica cómo configurar y probar un servidor EJB.

Configuración de un servidor EJB de región única

Esta sección indica cómo configurar un servidor EJB de CICS de región única. La región única es tanto una región de escucha como una AOR.

Esta configuración mínima se puede utilizar como base para el desarrollo de un servidor EJB de CICS multirregión, como se describe en "Configuración de un servidor EJB multirregión" en la página 266.

Importante

- Para mayor claridad, asumiremos que:
  1. Se empieza desde una región CICS Transaction Server para z/OS, Versión 4 Release 2 básica y sin personalizar.
  2. Habrá un único entorno de ejecución CorbaServer en el servidor EJB.
- Recomendamos que cuando cree el primer servidor EJB utilice el perfil de JVM predeterminado, DFHJVMCD. Cuando tenga su primer servidor EJB instalado y en ejecución, tal vez desee personalizar el perfil de JVM. En "Tras la ejecución del IVP de EJB: pasos opcionales" en la página 264 se describe cómo hacerlo.
- Esta sección no le indica cómo desplegar enterprise beans. El despliegue es un proceso independiente que se produce tras haber configurado el servidor EJB. Esto se describe en "Despliegue de enterprise beans" en la página 321.
- El resto de esta sección se divide en dos partes:
  - "Antes de ejecutar el procedimiento de verificación de instalación (IVP) de EJB" le lleva a poder ejecutar el programa de verificación de la instalación de EJB, que comprueba que ha configurado CICS correctamente como un servidor EJB y ha establecido un servidor de nombres de forma adecuada.

    Nota: De forma predeterminada, el IVP de EJB utiliza el COS Naming Server ligero `tnameserv` que se incluye con Java 1.3 y versiones posteriores. Por lo tanto, no es necesario configurar un servidor de nombres de calidad empresarial antes de ejecutar el IVP. Sin embargo, tras haber configurado el servidor de nombres "real", puede utilizar el IVP para probarlo.
  - "Tras la ejecución del IVP de EJB: pasos opcionales" en la página 264 describe algunas formas opcionales en las que puede personalizar el servidor EJB.

Antes de ejecutar el procedimiento de verificación de instalación (IVP) de EJB:

Los pasos de esta sección le permiten ejecutar el programa de verificación de la instalación de EJB, que comprueba que ha configurado CICS correctamente como un servidor EJB.

Los pasos de esta sección le permiten ejecutar el programa de verificación de la instalación de EJB, que comprueba que ha configurado CICS correctamente como un servidor EJB. Es necesario realizar acciones en:
1. z/OS o Windows NT, en función del tipo de servidor de nombres que utilice.
2. z/OS UNIX
3. CICS

Acciones necesarias en z/OS o Windows NT:

Para ejecutar el IVP de EJB, necesita un servidor de nombres que soporte la Java Naming and Directory Interface (JNDI) versión 1.2. De forma predeterminada, el IVP utiliza el COS Naming Server ligero `tnameserv` que se incluye con Java 1.3 y posteriores.

Para lanzar `tnameserv` en el host local, escriba el siguiente mandato en el indicador de mandatos de z/OS UNIX System Services o de Windows NT:

```
tnameserv -ORBInitialPort 2809
```

Esto hace que el servidor de nombres escuche las conexiones en el puerto TCP/IP 2809. Si este puerto ya está siendo utilizado por su sistema, se le pedirá que vuelva a intentarlo con un puerto distinto.

Nota: Si ejecuta software de cortafuegos, de forma predeterminada el cortafuegos puede bloquear el puerto especificado. Debe asegurarse de que la política de su cortafuegos permite que CICS y cualquier aplicación cliente de EJB se comuniquen con el servidor de nombres.

Para obtener información sobre cómo elegir y configurar un servidor de nombres con calidad empresarial, consulte “Habilitar referencias de JNDI” en la página 396.

Acciones necesarias en z/OS UNIX:

Para realizar las tareas que aparecen en esta sección, necesita un ID de usuario de z/OS UNIX con autorización de grabación en la vía de acceso del directorio que va a utilizar CICS.

Acerca de esta tarea

Cree los siguientes directorios en z/OS UNIX, si aún no existen. (Si ha configurado anteriormente CICS como un servidor IIOP, puede que algunos de estos directorios ya existan). Recuerde que los nombres de z/OS UNIX distinguen entre mayúsculas y minúsculas.

1. Un directorio de trabajo de CICS. Cada región CICS necesita un directorio de trabajo. El nombre lo especifica el parámetro WORK_DIR del perfil de JVM. Es necesario definir los permisos de directorio de forma que el ID de usuario con el que se ejecuta la región pueda leer y grabar en ese directorio. Consulte Concesión a las regiones CICS de acceso a z/OS UNIX System Services para obtener orientación.
2. Un directorio raíz de estantería. Puede llamar a su directorio de estantería como desee. Sin embargo, se recomienda crearlo en algún lugar bajo el directorio `/var`. Por ejemplo, puede crear un directorio de z/OS UNIX llamado `/var/cicsts/`. Tras haber creado el directorio de estantería, debe conceder al ID de usuario de la región CICS acceso completo al mismo: lectura, grabación y ejecución. En Concesión a las regiones CICS de acceso a z/OS UNIX System Services se describe cómo hacerlo.

3. Un directorio de archivo JAR desplegado (conocido también como directorio de recogida). Puede llamar a su directorio de recogida como desee. Sin embargo, se recomienda crearlo en algún lugar bajo el directorio `/var`. Por ejemplo, puede crear un directorio de z/OS UNIX llamado `/var/cicsts/pickup`. Debe conceder al ID de usuario de la región CICS, como mínimo, acceso de lectura a este.

   Nota:

   a. Si fuera a instalar varios entornos de ejecución de CorbaServer en el servidor EJB, tendría que crear un directorio de recogida distinto para cada uno.
   b. Si utiliza el mecanismo de exploración (para instalar archivos JAR desplegados desde el directorio de recogida) en una región de producción, tenga en cuenta las implicaciones de seguridad: específicamente, la posibilidad de que se omita la seguridad de mandatos CICS en definiciones. Para protegerse contra esto, recomendamos que los ID de usuario a los que se otorgue acceso de grabación al archivo JAR desplegado de z/OS UNIX se restrinjan a aquellos a los que se haya otorgado autoridad RACF para crear y actualizar definiciones DJAR y CORBASERVER.

Acciones necesarias en CICS:

Tenga en cuenta que si ha configurado anteriormente CICS como un servidor IIOP, para soportar llamada a método para objetos sin estado CORBA, tal vez ya haya completado algunos de estos pasos.

Acerca de esta tarea

1. Instale el IBM 64 bits SDK para z/OS, Java Technology Edition. Puede descargar este producto y obtener más información sobre el mismo en http://www.ibm.com/servers/eserver/zseries/software/java/.
2. Configure CICS para soporte de llamadas IIOP. (CICS utiliza el mismo protocolo de RMI sobre IIOP para el soporte de solicitudes de método de cliente para objetos sin estado CORBA y enterprise beans). En “Configuración de CICS para IIOP” en la página 409 se describe cómo hacerlo.

   Al leer “Configuración de CICS para IIOP” en la página 409, recuerde que:

   - Como nuestro servidor EJB de región única es una combinación de región de escucha/AOR, debe especificar 'YES' en el parámetro de inicialización del sistema IIOPLISTENER.
   - CICS carga perfiles de JVM desde el directorio de z/OS UNIX que se especifica mediante el parámetro de inicialización del sistema **JVMPROFILEDIR**. Asegúrese de que este valor especifica el directorio que contiene los perfiles de JVM que utilizar su región CICS.
   - Si desea utilizar su servidor de región única como base de un servidor multirregión, debe asegurarse de que el archivo de directorio de secuencias de solicitud, DFHEJDIR, y el archivo de almacén de objetos EJB, DFHEJOS, se pueden compartir en varias regiones. Por esta razón, se recomienda definirlos de una de las siguientes formas:
     – Como archivos remotos en una región propietaria del archivo (FOR)
     – Como tablas de datos de recursos de acoplamiento
     – Mediante el uso de compartición a nivel de registro VSAM (VSAM RLS).
   - Las definiciones PROGRAM no son necesarias para los enterprise beans como tales. Las únicas definiciones PROGRAM necesarias son aquellas para los programas receptor de peticiones y procesador de solicitudes. El

programa procesador de solicitudes predeterminado —indicado por la transacción predeterminada CIRP en las definiciones REQUESTMODEL— es DFJIIRP. CIRP y DFJIIRP se definen en el grupo de definición de recurso DFHIIOP provisto, al igual que CIRR y DFHIIRRS, la transacción y el programa receptor de peticiones. DFHIIOP se incluye en la lista de grupo inicial de CICS predeterminada.

Si utiliza un perfil de JVM distinto al predeterminado, DFHJVMCD, debe especificar el nombre del perfil en la opción JVMPROFILE de la definición PROGRAM para el programa procesador de solicitudes. (Es posible utilizar un mandato CEMT SET PROGRAM JVMPROFILE para cambiar el perfil de JVM del especificado en la definición PROGRAM instalada. Sin embargo, si crea su propio perfil de JVM, se recomienda crear nuevas definiciones TRANSACTION y PROGRAM para el programa procesador de solicitudes en lugar de cambiar las definiciones predeterminadas).

- Debe especificar la ubicación del servidor de nombres en la propiedad del sistema **-Dcom.ibm.cics.ejs.nameserver** de todos los perfiles que utilizan sus objetos CORBA sin estado o sus enterprise beans, incluidos los perfiles que CICS utiliza para publicar archivos JAR desplegados.

  Para obtener información detallada sobre la definición de la ubicación del servidor de nombres, consulte "Propiedades del sistema de la JVM" en la página 118.
- No es necesario instalar las definiciones REQUESTMODEL o DJAR en esta etapa, ya que:
  - Las aplicaciones de ejemplo EJB IVP y EJB utilizan el ID de transacción predeterminado de REQUESTMODEL, CIRP.
  - Es más fácil crear definiciones de REQUESTMODEL utilizando la transacción de CREA tras haber desplegado los enterprise beans en CICS. El despliegue es un proceso independiente que se produce tras haber configurado el servidor EJB. Esto se describe en "Despliegue de enterprise beans" en la página 321.
  - Las definiciones de DJAR generalmente las crea y las instala el mecanismo de exploración de CICS durante el despliegue.

3. Cree las siguientes definiciones de recurso de CICS:
   - Una TCPIPSERVICE
   - Una CORBASERVER

   El grupo de ejemplo proporcionado por CICS, DFH$EJB, contiene las definiciones TCPIPSERVICE y CORBASERVER adecuadas para ejecutar el EJB IVP. Debe modificar algunos de los atributos de estas definiciones de recurso para adecuarlas a su propio entorno. Para ello, utilice la transacción CEDA del programa de utilidad DFHCSDUP.

   a. Copie el grupo de ejemplo a un grupo de su elección. Por ejemplo:
      ```
      CEDA COPY GROUP(DFH$EJB) TO(mygroup)
      ```
   b. Visualice el grupo `mygroup` y cambie los siguientes atributos adecuadamente:
      - En la definición de recurso TCPIPSERVICE, modifique PORTNUMBER según sea necesario a un número de puerto TCP/IP de la instalación. El número de puerto que especifique debe ser autorizado por su administrador de red.

        Nota:

        1) Tenga en cuenta que en la definición TCPIPSERVICE que se proporciona:

- La opción PROTOCOL especifica IIOP. Este es el protocolo necesario para llamadas a método para enterprise beans y para objetos CORBA sin estado.
- La opción SSL especifica NO.
- La opción AUTHENTICATE es NO de forma predeterminado. Esto significa que el servicio de este puerto aceptará solicitudes IIOP de entrada sin autenticar.

2) Si desea utilizar su servidor de región única como base de un servidor multirregión, como se describe en "Configuración de un servidor EJB multirregión" en la página 266, debe especificar un valor para la opción DNSGROUP. Esto asegura que, en un servidor multirregión, podrá utilizar optimización de la conexión mediante registro de DNS dinámico para equilibrar las conexiones de cliente en las regiones de escucha.
3) Para obtener información de referencia sobre las definiciones de TCPIPSERVICE, consulte la *Guía de definición de recurso de CICS*.

- En la definición de recurso CORBASERVER:

1) Modifique la opción SHELF de manera que especifique el nombre completo del directorio de estantería de z/OS UNIX que se creara en el paso 2 de "Acciones necesarias en z/OS UNIX" en la página 259.

   Nota: En un servidor EJB multirregión, como la definición CORBASERVER se instalará en todas las AOR, este directorio de estantería de "alto nivel" lo compartirán todas ellas. Cada AOR creará automáticamente su propio subdirectorio bajo el directorio de estantería y un subdirectorio para el CorbaServer bajo este.
2) Modifique la opción DJARDIR de manera que especifique el nombre completo del directorio del archivo JAR desplegado de z/OS UNIX (directorio de recogida) que se creara en el paso 3 de "Acciones necesarias en z/OS UNIX" en la página 259.

   Nota: En un servidor EJB multirregión, el directorio de recogida (o los directorios, si las AOR contienen varios CorbaServers), al igual que el directorio de estantería, lo compartirán todas las AOR del servidor lógico.
3) Defina HOST como su nombre de host TCP/IP.

Nota:

1) Tenga en cuenta que en la definición CORBASERVER que se proporciona:
   - La opción UNAUTH especifica el nombre de la definición TCPIPSERVICE.

     Debe especificar siempre un valor para el atributo UNAUTH cuando defina un CorbaServer, aunque pretenda que se autentiquen todas las solicitudes de entrada a este CorbaServer. Esto se debe a que el número de puerto de TCPIPSERVICE se utiliza para construir Interoperable Object References (IOR) que se exportan desde este servidor lógico. Si especifica el nombre de otras definiciones de TCPIPSERVICE en las opciones CLIENTCERT o SSLUNAUTH, puede hacer que las regiones de escucha escuchen otros puertos para distintos tipos de solicitudes IIOP de entrada autenticadas. Para obtener más información, consulte la *Guía de definición de recurso de CICS*.

– La opción AUTOPUBLISH especifica YES. Esto hace que CICS publique beans en el espacio de nombres de forma automática cuando se instale una definición DJAR de forma correcta.
– La opción STATUS especifica Enabled.

2) El valor de la opción HOST de la definición CORBASERVER debe ser compatible con la de las opciones HOST o IPADDRESS correspondientes a los recursos TCPIPSERVICE asociados. En un servidor multirregión, si se utiliza registro de DNS dinámico para equilibrar las conexiones de cliente en las regiones de escucha, el valor de la opción HOST debe coincidir con el nombre de host genérico especificado en la opción DNSGROUP de la definición TCPIPSERVICE.
3) Para obtener información de referencia sobre las definiciones de CORBASERVER, consulte la *Guía de definición de recurso de CICS*.

c. Instale el grupo `mygroup` para informar de estas definiciones a CICS.

Cuando se instala la definición CORBASERVER, CICS:

1) Explora el directorio de recogida que se especificara en la opción DJARDIR.
2) Coapia los archivos JAR desplegados que encuentra en el directorio de recogida a su directorio de estantería.
3) Crea e instala de forma dinámica definiciones DJAR para los archivos JAR desplegados (si los hay) que encontrara en el directorio de recogida.
4) Debido a que la definición CORBASERVER especifica AUTOPUBLISH(YES), publica en el espacio de nombres de JNDI los enterprise beans contenidos en los DJAR.

d. Defina el estado de TCPIPSERVICE como OPEN:

```
CEMT SET TCPIPSERVICE(EJBTCP1) OPEN
```

En la CICS Console, debe ver, entre otros, mensajes similares al siguiente:

```
DFHEJ0701 Se ha creado CorbaServer EJB1.
DFHEJ5024 Comienzo de exploración para CorbaServer EJB1, el directorio que
          se explora es          nombre_DJARDIR.
DFHEJ5025 Exploración completada para CorbaServer EJB1, 0 DJar creados, 0 DJar
          actualizados.
DFHEJ1520 CorbaServer EJB1 ya es accesible.
DFHSO0107 TCPIPSERVICE EJBTCP1 se ha abierto en el puerto número_puerto, dirección
          IP xxx.xxx.xxx.xxx
```

donde:

- **nombre_DJARDIR** es el nombre del directorio ("de recogida") del archivo JAR desplegado del CorbaServer.
- **número_puerto** es el número del puerto TCP/IP utilizado por su CorbaServer.
- **xxx.xxx.xxx.xxx** es la dirección IP de su CorbaServer.

4. Configure CICS para utilizar JNDI. Para permitir que el código Java que se ejecuta en CICS emita llamadas a API de JNDI y que CICS publique referencias a las interfaces iniciales de enterprise beans, debe definir la ubicación del servidor de nombres. (En el caso de un servidor de nombres LDAP, es necesario especificar información adicional). Especifique el URL y el número de puerto del servidor de nombres en la propiedad del sistema **`-Dcom.ibm.cics.ejs.nameserver`**.

Por ejemplo, para utilizar `tnameserv`, el CORBA Object Services Naming Directory Server ligero incluido con Java 1.3 y posteriores, especifique:

```
-Dcom.ibm.cics.ejs.nameserver=iiop://tnameserv.yourcompany.com:2809
```

donde `tnameserv.yourcompany.com` es la dirección del host en el que se lanzó el servidor de nombres `tnameserv` y `2809` es el puerto que se seleccionó.

Si utiliza un servidor LDAP de calidad empresarial, tal vez especifique:

```
-Dcom.ibm.cics.ejs.nameserver=ldap://demojndi.yourcompany.com:389
```

Para conocer las demás propiedades que son necesarias y el modo de configurar su servidor de nombres LDAP, consulte "Configuración de un servidor LDAP" en la página 397.

Si utiliza un CORBA Object Services Naming Directory Server estándar, tal vez especifique:

```
-Dcom.ibm.cics.ejs.nameserver=iiop://demojndi.yourcompany.com:900
```

Si utiliza el CORBA Object Services Naming Directory Server incluido con WebSphere Application Server versión 5 o posteriores, debe especificar:

```
-Dcom.ibm.cics.ejs.nameserver=iiop://demojndi.yourcompany.com:2809/domain/legacyRoot
```

Importante: Para obtener información detallada sobre cómo definir la ubicación del servidor de nombres, consulte la descripción de la propiedad **-Dcom.ibm.cics.ejs.nameserver** en "Propiedades del sistema de la JVM" en la página 118.

El perfil de JVM para el programa procesador de solicitudes predeterminado es DFHJVMCD. Si ha seguido los pasos anteriores de esta sección, el perfil o los perfiles que está utilizando deberían estar en el directorio de z/OS UNIX especificado por el parámetro de inicialización del sistema **JVMPROFILEDIR**.

Importante: Estas instrucciones le han mostrado cómo configurar un servidor EJB de región única que contiene un único entorno de ejecución de CorbaServer. En una región de producción que soporte varias aplicaciones, cada una de las cuales utiliza su propio conjunto de enterprise beans, tal vez necesite varios CorbaServers. Para facilitar el mantenimiento en una región de producción, se deben seguir las pautas sobre cómo asignar beans a CorbaServers e ID de transacción que aparecen en "Actualización de enterprise beans en una región de producción" en la página 326.

Tras haber completado los pasos anteriores, puede, si lo desea, ejecutar el programa de verificación de la instalación de EJB, que comprueba que ha configurado CICS correctamente como un servidor EJB. Para obtener detalles sobre el IVP de EJB, consulte "Utilización del procedimiento de verificación de instalación (IVP) de EJB" en la página 276. Como alternativa, puede continuar con la siguiente sección antes de ejecutar el IVP.

Tras la ejecución del IVP de EJB: pasos opcionales:

Como opción, para finalizar la configuración de su servidor EJB completo, puede personalizar uno de los perfiles de JVM de ejemplo o crear sus propios perfiles de JVM para utilizarlos con enterprise beans, en lugar de emplear el perfil de JVM predeterminado DFHJVMCD.

Acerca de esta tarea

DFHJVMCD solo se puede personalizar de formas limitadas, ya que se utiliza para programas de CICS internos, pero otros perfiles de JVM se pueden personalizar tanto como se desee.

En "Configuración de JVM en agrupación" en la página 95 se indica cómo seleccionar y personalizar un perfil de JVM o, si lo prefiere, cómo crear su propio

perfil de JVM basado en uno de los perfiles de ejemplo que se brindan. Siga los procedimientos de dicha sección para personalizar o crear su propio perfil de JVM.

Cuando haya personalizado o creado su perfil de JVM, para que los enterprise beans utilicen el perfil:

1. Especifique el nombre del perfil de JVM en la opción JVMPROFILE de la definición PROGRAM para el programa procesador de solicitudes. (La definición PROGRAM que se proporciona para el programa procesador de solicitudes predeterminado, DFJIIRP, especifica el perfil predeterminado, DFHJVMCD).

   Debe crear sus propias definiciones TRANSACTION y PROGRAM para el programa procesador de solicitudes, como se describe en "Definición de recursos de CICS" en la página 411, en lugar de cambiar las definiciones predeterminadas. Especifique el nombre de su TRANSACTION en definiciones REQUESTMODEL para métodos de bean que se vayan a ejecutar en el nuevo perfil.
2. Coloque su perfil en el directorio de z/OS UNIX especificado por el parámetro de inicialización del sistema **JVMPROFILEDIR**.

Importante: Debe especificar la ubicación del servidor de nombres en la propiedad del sistema **-Dcom.ibm.cics.ejs.nameserver** en todos los perfiles de JVM o en los archivos de propiedades que sean utilizados por aplicaciones CORBA o enterprise beans, incluidos los perfiles que CICS utiliza para publicar archivos JAR desplegados. Para obtener información detallada sobre la definición de la ubicación del servidor de nombres, consulte "Propiedades del sistema de la JVM" en la página 118.

Prueba del servidor EJB

Esta sección indica cómo comprobar que el servidor EJB de CICS está configurado correctamente.

Ejecución del programa de verificación de la instalación (IVP) de EJB:

El modo más fácil de probar la configuración de EJB de CICS, incluida la del servidor de nombres, es ejecutar el programa de verificación de la instalación (IVP) de EJB incluido con CICS.

El IVP consta de:

- Un programa de cliente en modalidad de línea que se ejecuta en UNIX System Services (USS) en z/OS
- Un enterprise bean que se ejecuta en el servidor EJB de CICS.

Para ejecutar el IVP, debe haber completado todos los pasos que se muestran en "Antes de ejecutar el procedimiento de verificación de instalación (IVP) de EJB" en la página 258. Puede haber completado o no los pasos que aparecen en "Tras la ejecución del IVP de EJB: pasos opcionales" en la página 264. Ejecutar el IVP correctamente confirma que los programas externos son capaces de invocar enterprise beans en el servidor EJB de CICS.

Para obtener detalles sobre el IVP de EJB, consulte "Utilización del procedimiento de verificación de instalación (IVP) de EJB" en la página 276.

Utilización de la aplicación de ejemplo EJB "Hello World":

"Hello World" es una aplicación sencilla que consta de un formulario HTML, de un servlet Java y Java Server Pages que se ejecutan en un servidor web y de un enterprise bean de CICS.

Solicita al usuario una entrada, utiliza el enterprise bean para añadir la entrada del usuario a un mensaje estándar y luego visualiza la serie resultante.

Para ejecutar la aplicación de ejemplo EJB "Hello World", debe haber completado todos los pasos que se muestran en "Antes de ejecutar el procedimiento de verificación de instalación (IVP) de EJB" en la página 258. Puede haber completado o no los pasos que aparecen en "Tras la ejecución del IVP de EJB: pasos opcionales" en la página 264.

Para obtener detalles de la aplicación EJB "Hello World" e instrucciones sobre cómo instalarla, consulte "La aplicación de ejemplo "Hello World" de EJB" en la página 281.

Utilización de la aplicación de ejemplo EJB Bank Account:

Tras haber ejecutado la aplicación de ejemplo "Hello World" correctamente, tal vez desee intentar algo más ambicioso.

La aplicación de ejemplo EJB Bank Account demuestra cómo se puede utilizar un enterprise bean para hacer que información existente controlada por CICS esté disponible para los usuarios web. Extrae información sobre el cliente de tablas de datos y la devuelve al usuario.

La aplicación de ejemplo consta de un formulario HTML, un servlet Java y Java Server Pages que se ejecutan en un servidor web, un enterprise bean de CICS, dos programas de servidor COBOL de CICS y algunas tablas de datos de DB2. El enterprise bean utiliza el CCI Connector for CICS TS para enlazar con los programas de servidor de CICS con acceso a las tablas de datos de DB2.

Para ejecutar la aplicación de ejemplo EJB Bank Account, debe haber completado todos los pasos que se muestran en "Antes de ejecutar el procedimiento de verificación de instalación (IVP) de EJB" en la página 258. Puede haber completado o no los pasos que aparecen en "Tras la ejecución del IVP de EJB: pasos opcionales" en la página 264.

Para obtener detalles de la aplicación EJB Bank Account e instrucciones sobre cómo instalarla, consulte "La aplicación de ejemplo EJB Bank Account" en la página 290.

Utilización de sus propios enterprise beans:

Tras haber ejecutado las aplicaciones de ejemplo y haber establecido que su servidor EJB de CICS funciona correctamente, probablemente desee desplegar sus propios enterprise beans en CICS.

Para obtener detalles sobre cómo hacerlo, consulte "Despliegue de enterprise beans" en la página 321.

Configuración de un servidor EJB multirregión

Esta sección indica cómo configurar un servidor EJB lógico de CICS que conste de varias regiones de escucha y de varias AOR.

Antes de empezar

Para configurar un servidor EJB multirregión, debe haber creado ya un servidor EJB de región única como se describe en "Configuración de un servidor EJB de región única" en la página 258.

Acerca de esta tarea

Asegúrese de que las regiones de un servidor EJB multirregión, tanto de escucha como AOR, estén en el mismo nivel de CICS.

Procedimiento

1. Cree un conjunto de regiones de escucha mediante la clonación del CICS de servidor de región única. Todas las regiones clonadas comparten el archivo de definición de sistema CICS (CSD) del servidor de región única. Como opción, puede descartar las siguientes definiciones de recurso de las regiones de escucha, donde no son necesarias:
   - CORBASERVER
   - DJAR
   - DFHEJOS

   Deje el valor del parámetro de inicialización del sistema **IIOPLISTENER** definido como YES.

   Nota: Si se utiliza CICSPlex SM, se puede definir un grupo de CICS (CICSGRP) que contenga todas las regiones de escucha. Esto tiene la ventaja de que pueden asociarse recursos (mediante una descripción de recursos) con el grupo en lugar de con regiones individuales. Cuando se añade una región al grupo o se elimina de él, automáticamente los recursos se añaden a la región o se eliminan de ella.
2. Cree un conjunto de AOR mediante la clonación del CICS de servidor de región única. (Todas las regiones clonadas comparten el CSD del servidor de región única).

   Cada AOR debe utilizar el mismo contexto inicial de JNDI que las demás AOR.

   Como las AOR no son regiones de escucha, cambie el valor del parámetro de inicialización del sistema **IIOPLISTENER** a "NO".

   Nota: Si se utiliza CICSPlex SM, se puede definir un grupo de CICS (CICSGRP) que contenga todas las AOR. Cuando se añade una región al grupo o se elimina de él, automáticamente los recursos se añaden a la región o se eliminan de ella.

   La Figura 17 en la página 269 muestra qué definiciones son necesarias en las regiones de escucha, cuáles en las AOR y cuáles en ambas.
3. Conecte cada AOR a todas las regiones de escucha mediante MRO (no mediante ISC). Para obtener información sobre cómo definir conexiones MRO entre regiones de CICS, consulte la *Guía de intercomunicación de CICS*.

   Si se utiliza CICSPlex SM, se puede reducir de forma significativa el número de definiciones necesarias de CONNECTION y SESSION (y el coste de mantenerlas) definiendo SYSLINK desde una sola AOR a todas las regiones de escucha. (CICSPlex SM automáticamente crea las conexiones recíprocas desde las regiones de escucha a la AOR). Utilice los SYSLINK como modelos para las conexiones procedentes de las demás AOR.
4. Asegúrese de que el archivo del directorio EJB, DFHEJDIR, lo comparten todas las regiones del servidor EJB. Si definió DFHEJDIR en el servidor EJB de región única de la forma sugerida (es decir, como un archivo remoto, una tabla de

datos de recurso de acoplamiento o utilizando VSAM RLS), automáticamente el archivo deben compartirlo entre las regiones clonadas del servidor multirregión.

Nota: Asegúrese de que la región CICS que es propietaria del archivo DFHEJDIR se inicia antes de que las demás regiones tengan acceso al mismo, especialmente las AOR. Si no lo hace, los intentos de instalar definiciones CORBASERVER y DJAR en las otras AOR no tendrán éxito y se generará el mensaje DFHEJ0736.

5. Asegúrese de que el archivo del almacén de objetos EJB, DFHEJOS, lo comparten todas las regiones del servidor EJB. Si definió DFHEJOS en el servidor EJB de región única de la forma sugerida, el archivo deben compartirlo automáticamente todas las regiones clonadas del servidor multirregión. (Como opción, puede suprimir la definición DFHEJOS de las regiones de escucha, donde no es necesaria).
6. Para equilibrar las conexiones de cliente en las regiones de escucha, utilice optimización de conexiones mediante el registro de DNS: En "Optimización de la conexión mediante Sistema de nombres de dominio (DNS)" en la página 389 se describe cómo configurarla.
7. Disponga que las solicitudes de método para enterprise beans se dirijan de forma dinámica a través de las AOR. Puede utilizar cualquiera de las siguientes opciones:
   a. CICSPlex SM. En "CICSPlex SM con enterprise beans" en la página 376 se describe cómo utilizar CICSPlex SM para dirigir solicitudes de método para enterprise beans.
   b. Una versión personalizada del programa de direccionamiento distribuido de CICS, DFHDSRP. En la *Guía de personalización de CICS* se describe cómo escribir un programa de direccionamiento distribuido para dirigir solicitudes de método para enterprise beans y objetos CORBA sin estado.

   En el parámetro de inicialización del sistema DSRTPGM para las regiones de escucha, especifique el nombre del programa de direccionamiento distribuido que hay que utilizar. Si está utilizando CICSPlex SM, especifique el nombre del programa de direccionamiento de CICSPlex SM, EYU9XLOP. En caso contrario, especifique el nombre del programa de direccionamiento personalizado. Para obtener más información sobre el parámetro de inicialización del sistema DSRTPGM , consulte Parámetro de inicialización del sistema DSRTPGM en la Guía de definición de sistema.

   Recuerde:

   a. Para dirigir solicitudes de método para enterprise beans de forma dinámica, la definición TRANSACTION para la transacción indicada en las definiciones de REQUESTMODEL debe especificar DYNAMIC(YES). La transacción predeterminada indicada en definiciones REQUESTMODEL, CIRP, se define como DYNAMIC(NO). Recomendamos realizar una copia de la definición TRANSACTION para CIRP, cambiar el valor de DYNAMIC y guardar la definición con un nuevo nombre. Luego indique el nombre de su nueva transacción en las definiciones REQUESTMODEL. (La forma más fácil de crear definiciones REQUESTMODEL es utilizar la transacción CREA tras haber desplegado los enterprise beans en CICS).
   b. La definición de transacción "común" especificada en el parámetro de inicialización del sistema DTRTRAN y utilizada para solicitudes de direccionamiento de transacción iniciado por un terminal si no se encuentra ninguna definición TRANSACTION, nunca se asocia con solicitudes de método para enterprise beans. Si en la región de escucha no hay ninguna

definición REQUESTMODEL que coincida con la solicitud, esta se ejecuta en la transacción CIRP (que especifica DYNAMIC(NO)).

c. En la Figura 17, las definiciones REQUESTMODEL de las AOR son necesarias para las solicitudes salientes a objetos locales. Si un objeto CORBA sin estado o un enterprise hace una llamada a otro objeto y dicho objeto está disponible en la AOR local, CICS no envía la solicitud a una región de escucha. En su lugar, ejecuta el método llamado en la tarea actual ("bucle de retorno hermético") o inicia otro procesador de solicitudes en la AOR local ("bucle de retorno normal"). Si se utiliza el bucle de retorno normal, es preferible que la nueva tarea del procesador de solicitudes utilice la misma REQUESTMODEL que se utiliza para llamar al primer objeto o pueden producirse resultados imprevisibles. Si los objetos CORBA sin estado y los enterprise beans no realizan llamadas salientes, las REQUESTMODEL de la AOR no son estrictamente necesarias.

Resultados

Estos pasos describen cómo configurar un servidor EJB multirregión en el que cada región contiene un único entorno de ejecución de CorbaServer. En regiones de producción que soporten varias aplicaciones, cada una de las cuales utilice su propio conjunto de enterprise beans, tal vez necesite varios CorbaServers. Para facilitar el mantenimiento en regiones de producción, siga las pautas sobre cómo asignar beans a CorbaServers e ID de transacción que aparecen en "Actualización de enterprise beans en una región de producción" en la página 326.

Este diagrama muestra qué definiciones son necesarias en las regiones de escucha, cuáles en las AOR y cuáles en ambas.

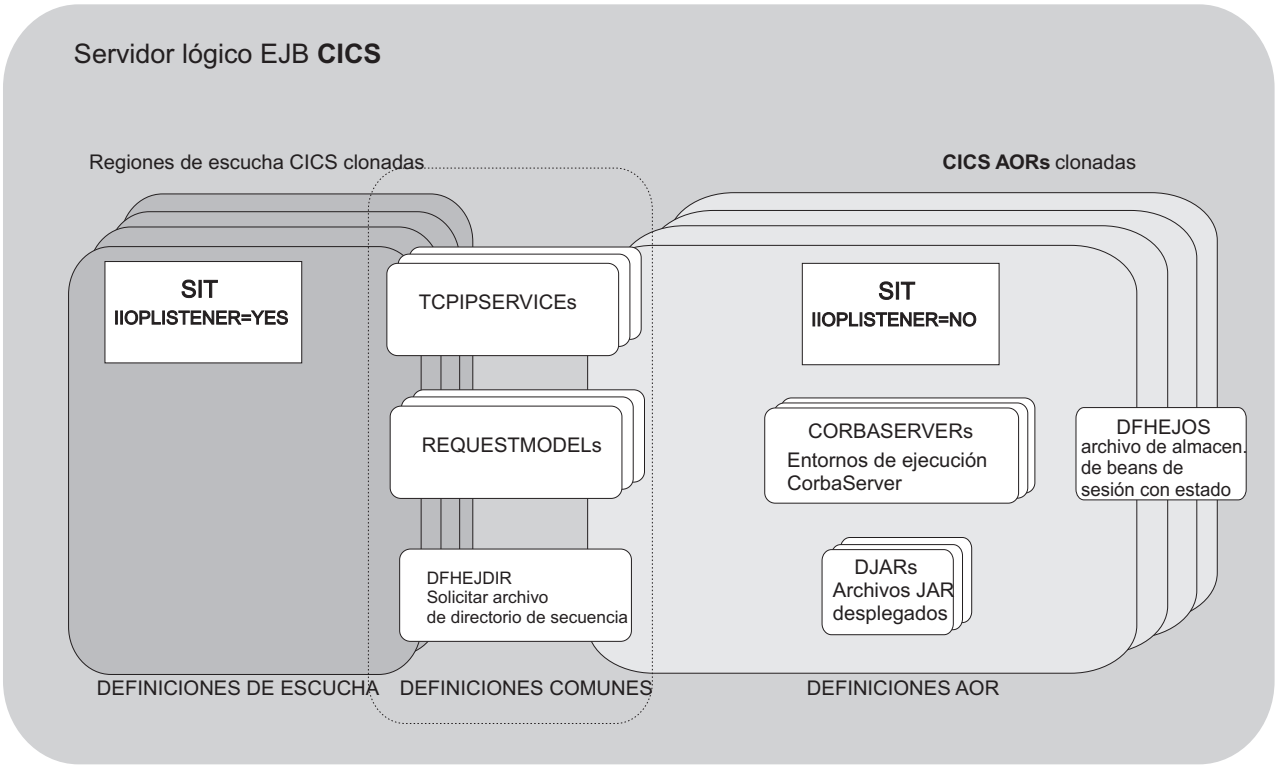


Figura 17. Definiciones de recurso en un servidor EJB de CICS multirregión

Actualización de un servidor EJB a CICS Transaction Server para z/OS, Versión 4 Release 2

Esta sección indica cómo actualizar un servidor EJB de nivel anterior a CICS TS para z/OS, Versión 4.2.

Actualización de un servidor EJB/CORBA de CICS de región única:

Realice estos pasos para actualizar un servidor EJB/CORBA de CICS de región única a CICS Transaction Server para z/OS, Versión 4 Release 2.

Procedimiento

1. Desactive temporalmente la carga de trabajo.
2. Apague la región.
3. Actualice esta región a CICS Transaction Server para z/OS, Versión 4 Release 2, siguiendo los procedimientos de actualización estándares descritos en la información de actualización definida para la versión desde la que actualiza.
4. Revise "Sugerencia de actualización" en la página 275, que describe algunos de los cambios en el soporte de EJB/CORBA entre distintas versiones de CICS. También puede consultar "Configuración de un servidor EJB de región única" en la página 258, que describe con detalle cómo configurar un servidor EJB de región única en CICS TS para z/OS, Versión 4.2.
5. Reinicie la región.
6. Vuelva a publicar las Interoperable Object Reference (IOR) para todos los enterprise beans y los objetos CORBA sin estado procesados por el servidor emitiendo un mandato **PERFORM CORBASERVER**(*nombre_CorbaServer*) PUBLISH. Puede emitir este mandato mediante **EXEC CICS**, CEMT, mediante el Gestor de recursos para enterprise beans o mediante la vista WUI de CICSPlex SM. Recuerde emitir un mandato distinto para cada CorbaServer de la región.

Actualización de un servidor EJB/CORBA de CICS multirregión:

Para actualizar un servidor EJB/CORBA de CICS multirregión a CICS Transaction Server para z/OS, Versión 4 Release 2, puede utilizar cualquiera de los métodos siguientes.

Acerca de esta tarea

1. **Concluya el servidor, actualice todas las regiones y reinicie el servidor.**

   Este método es muy similar al que se describe en "Actualización de un servidor EJB/CORBA de CICS de región única", excepto que:

   a. Debe actualizar todas regiones a CICS Transaction Server para z/OS, Versión 4 Release 2 antes de reiniciar el servidor. De nuevo, siga los procedimientos de actualización estándares descritos en la información de actualización definida para la versión desde la que actualiza.
   b. Debería consultar "Configuración de un servidor EJB multirregión" en la página 266, que describe con detalle cómo configurar un servidor multirregión en CICS TS para z/OS, Versión 4.2.
   c. Para volver a publicar las IOR de los enterprise beans y los objetos CORBA sin estado, emita un mandato PERFORM CORBASERVER(*nombre_CorbaServer*) PUBLISH en al menos una de las AOR. Recuerde emitir un mandato distinto para cada CorbaServer de la AOR.

   La ventaja de este método es su sencillez relativa, en comparación con las soluciones 2 y 3. Su desventaja principal es que las aplicaciones del servidor no están disponibles durante el proceso de actualización.

2. **Cree un servidor lógico CICS TS para z/OS, Versión 4.2, distinto y mueva las aplicaciones gradualmente desde el servidor antiguo del nivel anterior al nuevo.**

   Las ventajas de este método son:

   a. Las aplicaciones siguen estando disponibles durante el proceso de actualización.
   b. Puede empezar con un servidor CICS TS para z/OS, Versión 4.2 mínimo, que quizá conste de tan solo dos regiones: una región de escucha y una AOR. A medida que se muevan más aplicaciones, puede expandir el servidor CICS TS para z/OS, Versión 4.2 y, a la vez, reducir el número de regiones en el servidor de nivel anterior, y de ese modo conservará recursos.
   c. Probablemente sea más fácil de implementar que la solución 3.

   Para configurar un nuevo servidor EJB multirregión de CICS TS para z/OS, Versión 4.2, siga todos los pasos de "Configuración de un servidor EJB de región única" en la página 258 y de "Configuración de un servidor EJB multirregión" en la página 266.
3. **Realice una "actualización gradual"**.

   En una "actualización gradual", se actualiza una región cada vez desde el nivel anterior de CICS al actual, mientras el servidor se mantiene operativo.

   Las ventajas de este método son:

   a. Las aplicaciones siguen estando disponibles durante el proceso de actualización.
   b. A diferencia de la solución 2, en ninguna fase es necesario configurar regiones CICS adicionales.

   Este método se describe con detalle en "Realización de una "actualización gradual"".

Realización de una "actualización gradual":

El nivel mixto de operación que se describe en esta sección, en el que distintas regiones de CICS del mismo servidor lógico están en niveles distintos de CICS, está concebido solo para su uso en actualizaciones graduales.

Importante

No debe utilizarse de forma permanente, ya que esto aumenta el riesgo de error en algunos escenarios de interoperatividad. La modalidad de operación normal y recomendada es que todas las regiones de un servidor lógico deben estar en el mismo nivel de CICS y Java.

Esta sección describe cómo realizar una "actualización gradual" de un servidor EJB/CORBA de CICS multirregión a CICS Transaction Server para z/OS, Versión 4 Release 2. El proceso consta de los siguientes pasos:

1. Comprobación de que el servidor lógico cumple los criterios para una "actualización gradual". Consulte "Requisito".
2. "Pasos preliminares" en la página 272
3. "Actualización de las regiones de escucha" en la página 272
4. "Actualización de las AOR" en la página 273
5. "Últimas tareas" en la página 275

Requisito:

Su servidor debe constar de regiones de escucha y de regiones propietarias de la aplicación independientes. Esto se debe a que el proceso de actualización necesita que todas las regiones de escucha se actualicen antes que cualquier región propietaria de la aplicación (AOR).

Si se ejecutan regiones compuestas de región de escucha-AOR que puedan actuar de receptores de peticiones y de procesadores de solicitudes, esto no se puede realizar. Si no actualiza todas las regiones de escucha antes de actualizar cualquier AOR, las aplicaciones cliente de IIOP pueden recibir anomalías transitorias durante la ventana de migración, en función de la versión de CICS de la región de escucha que reciba la solicitud.

Pasos preliminares:
Acerca de esta tarea

1. Revise "Sugerencia de actualización" en la página 275.
2. Si actualiza desde CICS TS **2.2**, asegúrese de que el APAR PQ 79565 esté instalado en todas las regiones CICS TS 2.2. Este APAR mejora los diagnósticos de CICS TS 2.2, si hay carga de trabajo de CICS TS para z/OS, Versión 4.2 que llega a una región CICS TS 2.2. También permite que un procesador de solicitudes de CICS TS 2.2 (AOR) reciba trabajo de un receptor de peticiones (región de escucha) de CICS TS para z/OS, Versión 4.2.
3. Defina la opción AUTOPUBLISH en todas las definiciones CORBASERVER a `NO`. Definir un CorbaServer para que publique automáticamente las IOR en los espacios de nombres JNDI podría alterar el proceso de actualización.
4. Si utiliza un programa de direccionamiento distribuido para equilibrar las solicitudes de método para enterprise beans y objetos sin estado CORBA en las AOR del servidor lógico, personalice el programa de direccionamiento para utilizar el parámetro DYRLEVEL. DYRLEVEL es una ayuda para la actualización. Contiene el nivel de CICS necesario en la AOR de destino para procesar correctamente la solicitud dirigida. (Tenga en cuenta que este es el nivel de CICS **específico**, *no* el nivel mínimo, necesario para el proceso correcto de la solicitud). En un nivel mixto, cuando se invoca el programa de direccionamiento para la selección de ruta (o para error de selección de ruta), puede utilizar el valor de DYRLEVEL para determinar si dirigir la solicitud a una AOR de nivel anterior o CICS TS para z/OS, Versión 4.2.

   Para obtener detalles de cómo utilizar DYRLEVEL e información definitiva sobre cómo escribir un programa de direccionamiento distribuido, consulte la *Guía de personalización de CICS*.

   Instale el programa personalizado en *todas* las regiones (tanto de escucha como AOR) del servidor lógico.

   Si utiliza CICSPlex SM para equilibrar la carga de trabajo de sus solicitudes de método, puede saltarse este paso. El programa de direccionamiento de CICSPlex SM proporcionado con CICS Transaction Server para z/OS, Versión 4 Release 2 comprueba el campo DYRLEVEL y dirige las solicitudes de acuerdo con ello.

Actualización de las regiones de escucha:

Realice estos pasos para actualizar una región de escucha.

Acerca de esta tarea

1. Desactive temporalmente una región de escucha y apáguela.

2. Actualice esta única región de escucha a CICS Transaction Server para z/OS, Versión 4 Release 2, siguiendo los procedimientos de actualización estándares descritos en la información de actualización definida para la versión desde la que actualiza.

   Importante:

   a. Si se actualiza un CSD de CICS TS **2.2** al nivel CICS TS para z/OS, Versión 4.2, si este lo comparte cualquier región CICS TS 2.2 que no sea de las que se actualizan, incluya el grupo de recursos DFHCOMPA (incluido con CICS TS para z/OS, Versión 4.2) en la lista de grupos iniciales para estas regiones. DFHCOMPA es un grupo de compatibilidad que proporciona una definición de DFJIIRP, el programa procesador de solicitudes predeterminado, que puede utilizar una región CICS TS 2.2 al compartir un CSD CICS TS para z/OS, Versión 4.2.

      Este paso es necesario porque, en CICS TS para z/OS, Versión 4.2, el perfil de JVM utilizado por DFJIIRP es DFHJVMCD. En CICS TS 2.2 es DFHJVMPR.

   b. En esta etapa, no habilite ninguna nueva opción específica de CICS TS para z/OS, Versión 4.2 en definiciones de recurso, ya que las AOR de nivel anterior no las comprenderán. El uso de estas nuevas característica debe esperar hasta que se haya actualizado todo el servidor lógico, tanto las regiones de escucha como las AOR.

   Para obtener información definitiva sobre cómo configurar una región de escucha en CICS TS para z/OS, Versión 4.2, consulte “Configuración de CICS para IIOP” en la página 394.
3. Reactive la región de escucha. Ahora esta región está en la versión más nueva de CICS, pero debe seguir participando como parte del servidor lógico de nivel anterior.
4. Repita los pasos del 1 al 3 para todas las regiones de escucha que haya en el servidor lógico.

Actualización de las AOR:

Para actualizar una AOR para enterprise beans, realice estos pasos.

Acerca de esta tarea

1. Desactive temporalmente una AOR y apáguela.
2. Actualice esta única AOR a CICS Transaction Server para z/OS, Versión 4 Release 2, siguiendo los procedimientos de actualización estándares descritos en la información de actualización definida para la versión desde la que actualiza.

   Si actualiza desde CICS TS 2.2, parte de esto implica actualizar el perfil de JVM utilizado por los CorbaServer. Tenga en cuenta los cambios en los perfiles de JVM y los archivos de propiedades que se introdujeron en CICS TS 2.3, como se describe en “Sugerencia de actualización” en la página 275.

   Importante:

   a. Si se actualiza un CSD de CICS TS **2.2** al nivel CICS TS para z/OS, Versión 4.2, si este lo comparte cualquier región CICS TS 2.2 que no sea de las que se actualizan, incluya el grupo de recursos DFHCOMPA (incluido con CICS TS para z/OS, Versión 4.2) en la lista de grupos iniciales para estas regiones.

   b. En esta etapa, no habilite ninguna nueva opción específica de CICS TS para z/OS, Versión 4.2 en definiciones de recurso.

3. Reactive la AOR.
4. Asegúrese de que todos los TCPIPSERVICE están abiertos tanto en esta AOR como en las regiones de escucha.
5. Utilice el mandato CEMT PERFORM DJAR PUBLISH para volver a publicar las IOR de uno o más enterprise beans en formato de CICS TS para z/OS, Versión 4.2. Para cada CorbaServer, seleccione uno o más archivos JAR desplegados para volver a publicar. Al elegir archivos JAR desplegados para volver a publicar, tenga en cuenta lo siguiente:
   - Intente elegir DJAR cuya carga de trabajo completa pueda procesarla una única región.
   - Siempre que sea posible, todos los beans utilizados por una aplicación deben actualizarse a la vez. Por ejemplo, si se sabe que el bean A llama al bean B, los dos beans deben actualizarse a la vez. Si esto no es posible, el bean A debe actualizarse antes.

     Esto es especialmente importante si se actualiza desde CICS TS 2.2 y los beans se instalan en el mismo CorbaServer pero en AOR distintas que estén en niveles de CICS distintos. Esto se debe a que una región CICS TS 2.2 no puede realizar una búsqueda JNDI de un objeto en una región CICS TS para z/OS, Versión 4.2 si ambos objetos están en el mismo CorbaServer. Por ejemplo, el bean A en el CorbaServer EJB1 de una AOR CICS TS 2.2 no puede buscar el bean B en el CorbaServer EJB1 en una AOR CICS TS para z/OS, Versión 4.2.

     Nota: Si los beans A y B están instalados en CorbaServer distintos, o en AOR que estén en el mismo nivel de CICS, pueden actualizarse por separado.

   Vuelva a publicar los DJAR seleccionados en el espacio de nombres de JNDI, en la misma ubicación que la que utilizan las AOR de nivel anterior.

   En este punto :
   - Esta AOR está lista para aceptar carga de trabajo.
   - El servidor lógico contiene una agrupación de AOR de nivel anterior y una agrupación (que actualmente contiene una única región) de AOR CICS TS para z/OS, Versión 4.2.
   - Cualquier cliente que busque la IOR de un bean vuelto a publicar en el espacio de nombres obtiene la nueva IOR en formato CICS TS para z/OS, Versión 4.2. El programa de direccionamiento personalizado o CICSPlex SM dirige dichas solicitudes a la AOR CICS TS para z/OS, Versión 4.2.
   - Cualquier cliente que tenga una IOR obsoleta almacenada en memoria caché para un bean que se haya vuelto a publicar todavía puede utilizar dicho bean. El programa de direccionamiento personalizado o CICSPlex SM dirige dichas solicitudes de formato antiguo a una de las AOR de nivel anterior.

     Nota: Muchos servidores de aplicaciones copian en caché los resultados de las búsquedas JNDI de forma local para mejorar el rendimiento, por lo que puede descubrir que estas memorias caché se han depurado antes de utilizar las nuevas IOR. Durante un periodo de tiempo, las solicitud de enterprise beans vueltos a publicar se deben mover gradualmente de la agrupación de AOR de nivel anterior a la agrupación de AOR CICS TS para z/OS, Versión 4.2.
6. Repita los pasos del 1 al 5 para todas las AOR que haya en el servidor lógico. A medida que se actualiza cada AOR:
   - Vuelva a publicar un conjunto de enterprise beans, de forma que cada vez más y más beans tengan soporte de la agrupación de regiones CICS TS para z/OS, Versión 4.2.

- Cada vez es menos importante a la hora de seleccionar archivos JAR desplegados para volver a publicar elegir aquellos cuya carga de trabajo completa se pueda procesar mediante una única región, ya que hay más AOR en la agrupación de CICS TS para z/OS, Versión 4.2.

Finalmente, todas las AOR se ejecutarán en CICS TS para z/OS, Versión 4.2 y procesarán el 100% de la carga de trabajo.

Últimas tareas:

Para completar la actualización gradual, debe realizar estas tareas finales.

Acerca de esta tarea

Procedimiento

1. Si es necesario, restablezca la opción AUTOPUBLISH en las definiciones CORBASERVER a YES.
2. Habilite cualquier opción de definición de recurso específica de CICS TS para z/OS, Versión 4.2 que desee utilizar.

Resultados

Sugerencia de actualización:

Esta sección lista brevemente algunas sugerencias generales, como recordatorio de las cosas que hay que saber al actualizar un servidor EJB a CICS TS para z/OS, Versión 4.2.

Todos estos cambios se describen con detalle en Capítulo 4, “Configuración del soporte de Java”, en la página 81.

1. Los perfiles de JVM se almacenan en el directorio de z/OS UNIX al que apunta el parámetro de inicialización del sistema **JVMPROFILEDIR**.
2. El perfil de JVM predeterminado que utilizan los CorbaServer es DFHJVMCD.
3. No habilite ningún atributo nuevo específico de CICS TS para z/OS, Versión 4.2 en definiciones de recurso durante el proceso de “actualización gradual”. El uso de estas nuevas característica debe esperar hasta que se haya actualizado todo el servidor lógico, tanto las regiones de escucha como las AOR.
4. Desde una AOR CICS TS para z/OS, Versión 4.2, puede volver a publicar un archivo JAR desplegado que se haya publicado anteriormente desde una versión anterior de CICS sin retirarlo antes. Las IOR de los beans se actualizan al formato para la nueva versión. **Sin embargo, no puede hacer lo contrario**. Desde una versión anterior de CICS, antes de volver a publicar un archivo JAR desplegado que ya se haya publicado antes desde una AOR CICS TS para z/OS, Versión 4.2, primero debe retirarlo; además, como las versiones anteriores de CICS no entienden el formato de las IOR CICS TS para z/OS, Versión 4.2, *debe retirarlo de una AOR CICS TS para z/OS, Versión 4.2*.

   Recuerde esto si, por cualquier razón, tiene que retrotraer la actualización de una o más AOR. Si alguna vez tiene que revertir las IOR de enterprise beans que se hayan publicado desde una AOR CICS TS para z/OS, Versión 4.2 a un nivel anterior de CICS (para que se puedan dirigir a una AOR de nivel anterior una vez más), debe hacer lo siguiente:

   a. Retire el archivo JAR desplegado desde una AOR CICS TS para z/OS, Versión 4.2.
   b. Publique el archivo JAR desplegado desde una AOR de nivel anterior.

Si intenta volver a publicar los beans sin retirarlos antes, o si intenta retirarlos desde el nivel equivocado de CICS, se produce una excepción `InvalidUserKeyException: número de versión erróneo`.

Problemas potenciales:

1. Tras haber actualizado el servidor EJB a CICS TS para z/OS, Versión 4.2, algunos clientes pueden tener IOR obsoletas almacenadas en memoria caché que apunten al servidor antiguo. Esto se debe a que muchos servidores de aplicaciones copian en caché los resultados de las búsquedas JNDI de forma local para mejorar el rendimiento. Tal vez descubra que es necesario depurar estas memorias caché antes de utilizar las nuevas IOR.
2. CICS TS 2.3 y posteriores, incluido CICS TS para z/OS, Versión 4.2, soportan GIOP 1.2, mientras que CICS TS 2.2 únicamente soporta GIOP 1.1. Si se recibe un mensaje de GIOP 1.2 en una región CICS TS 2.2, se rechazará. En condiciones normales esto no debería suceder nunca, porque la versión máxima de GIOP soportada por CICS se almacena en las IOR que publica CICS. Si un cliente sabe que un servidor determinado solo soporta GIOP 1.1, nunca intentará utilizar otra versión más reciente al comunicarse con ese servidor. Esto significa que CICS TS para z/OS, Versión 4.2 puede enviar mensajes de GIOP a CICS TS 2.2.

   El problema solo se producirá si el cliente cree que habla con CICS TS para z/OS, Versión 4.2 (o con CICS TS 3.1 o CICS TS 2.3) pero este mensaje se dirige a una región CICS TS 2.2. Esto se producirá únicamente si las regiones CICS TS 2.2 y CICS TS para z/OS, Versión 4.2 se configuran como procesadores de solicitudes hermanos (AOR) en el mismo servidor lógico. (Esta es una de las razones por las que no se recomiendan servidores lógicos de nivel mezclado en CICS). Durante una "actualización gradual", el servidor lógico, por supuesto, contiene procesadores de solicitudes de nivel mezclado. Sin embargo, si se siguen los pasos de "Realización de una "actualización gradual"" en la página 271, el problema (de que se reciba un mensaje de GIOP 1.2 en una región CICS TS 2.2) no se producirá.
3. CICS TS 2.3 y posteriores, incluido CICS TS para z/OS, Versión 4.2, utilizan un formato de IOR distinto a CICS TS 2.2. Si un mensaje de GIOP 1.1 dirigido a CICS TS para z/OS, Versión 4.2 se dirige a una región CICS TS 2.2, la región CICS TS 2.2 rechazará la solicitud debido al uso de un formato de IOR desconocido. Si todas las regiones de un servidor EJB/CORBA están en el mismo nivel de CICS y Java, este error no se puede producir.

   Durante una "actualización gradual", el servidor lógico, por supuesto, contiene regiones de nivel mezclado. Sin embargo, si se siguen los pasos de "Realización de una "actualización gradual"" en la página 271, este problema no se producirá.

Utilización del procedimiento de verificación de instalación (IVP) de EJB

El programa de verificación de instalación (IVP) de EJB es una pequeña aplicación que el instalador de CICS puede utilizar para verificar el entorno de EJB de CICS.

El IVP de EJB utiliza un programa cliente que no requiere el uso de un servidor web. El IVP consta de:

- Un programa cliente de modalidad de línea que se ejecutar en UNIX System Services en z/OS.
- Un enterprise bean de sesión sin estado que se ejecuta en el servidor EJB de CICS.

El IVP prueba:
- La JVM de CICS (incluida su posibilidad de reutilización).
- Como opción, el servidor de nombres "real" de nivel de empresa. (De forma predeterminada, el IVP utiliza el COS Naming Server `tnameserv` ligero proporcionado con Java).
- La capacidad del servidor EJB de ejecutar un enterprise bean básico.
- Los valores de z/OS UNIX (incluidos los permisos de acceso a archivos).

Una vez configurado, el cliente:
1. Realiza una búsqueda JNDI para encontrar la referencia publicada a un enterprise bean específico en el espacio de nombres de JNDI.
2. Crea una nueva instancia del enterprise bean en CICS.
3. Llama a un método remoto en la instancia de bean.

Requisitos previos para el procedimiento de verificación de instalación (IVP) de EJB

Antes de ejecutar el IVP de EJB, necesitará estos recursos.
- Un ID de usuario y un editor de archivos de UNIX System Services.
- Un servidor EJB de CICS. La manera de configurar uno se describe en "Configuración de un servidor EJB de región única" en la página 258.
- Un servidor de nombres que soporte la Java Naming and Directory Interface (JNDI) versión 1.2 o posteriores. La forma de configurar un servidor de nombres con calidad empresarial se describe en "Habilitar referencias de JNDI" en la página 396. Como alternativa, puede utilizar el COS Naming Server `tnameserv` ligero proporcionado con Java.

Nota:

1. Estos requisitos previos asumen que se está probando un servidor EJB de CICS de región única.
2. Para ejecutar el IVP, debe haber completado los pasos que se muestran en "Antes de ejecutar el procedimiento de verificación de instalación (IVP) de EJB" en la página 258.
3. Antes de empezar, compruebe que el tamaño de almacenamiento para la sesión de TSO es de al menos 6.000 KB. Para aumentar el tamaño de almacenamiento, en la pantalla de inicio de sesión estándar de TSO cambie el valor del campo SIZE (Tamaño).

Instalación del procedimiento de verificación de instalación (IVP) de EJB

Para instalar EJB debe configurar z/OS UNIX y CICS. En z/OS UNIX System Services, debe configurar el cliente.

Configuración de z/OS UNIX para el IVP de EJB:

El IVP utiliza el mismo enterprise bean de CICS que la aplicación de ejemplo "Hello World" de EJB.

La aplicación de ejemplo se describe en "La aplicación de ejemplo "Hello World" de EJB" en la página 281. Por lo tanto, en z/OS UNIX debe copiar el archivo JAR desplegado `HelloWorldEJB.jar` del directorio de ejemplos de EJB al directorio del archivo JAR desplegado que se creara en "Antes de ejecutar el procedimiento de verificación de instalación (IVP) de EJB" en la página 258.

Nota: Tanto el código fuente como el código ejecutable del enterprise bean están en el archivo `HelloWorldEJB.jar`.

El directorio de ejemplos es: /usr/lpp/cicsts/cicsts42/samples/ejb/helloworld, donde /usr/lpp/cicsts/cicsts42 es el directorio de instalación para archivos de CICS en z/OS UNIX.

Recuerde que los nombres de z/OS UNIX distinguen entre mayúsculas y minúsculas.

Configuración de CICS:

Antes de ejecutar el IVP de EJB, debe realizar estas tareas de configuración de CICS.

Acerca de esta tarea

1. Si la seguridad basada en roles de EJB está activa en su región CICS, debe desactivarla antes de ejecutar el IVP. Es decir, si ambos parámetros de inicialización del sistema SEC y XEJB actualmente especifican 'YES', debe definir XEJB como 'NO' y reiniciar CICS.
2. El grupo de recursos de ejemplo proporcionado por CICS, DFH$EJB, contiene las definiciones TCPIPSERVICE y CORBASERVER adecuadas para ejecutar el IVP. Debe modificar algunos de los atributos de estas definiciones de recurso para adecuarlas a su propio entorno e instalar las definiciones cambiadas en CICS. Ya debería de haber hecho esto, como parte de la tarea de configurar el servidor EJB. Si no lo ha hecho, siga las instrucciones paso a paso que se muestran en “Acciones necesarias en CICS” en la página 260.
3. Emita un mandato `CEMT PERFORM CORBASERVER(EJB1) SCAN`.

   CICS:

   a. Explora el directorio de recogida que se especificara en la opción DJARDIR de la definición de CORBASERVER.
   b. Copia el archivo JAR desplegado `HelloWorldEJB.jar` que encuentra en el directorio de recogida en su directorio de estantería.
   c. Crea e instala dinámicamente una definición de DJAR para `HelloWorldEJB.jar`.
   d. Debido a que la definición CORBASERVER especifica AUTOPUBLISH(YES), publica el enterprise bean contenido en `HelloWorldEJB.jar` en el espacio de nombres de JNDI.
4. Si no lo ha hecho al configurar el CorbaServer, defina el estado de TCPIPSERVICE como OPEN:

   ```
   CEMT SET TCPIPSERVICE(EJBTCP1) OPEN
   ```

   En la CICS Console, debe ver, entre otros, mensajes similares al siguiente:

   ```
   DFHEJ5024 Comienzo de exploración para CorbaServer EJB1, el directorio que
             se explora es          nombre_DJARDIR.
   DFHEJ5030 Se está creando el nuevo DJar HelloWorldEJB durante una exploración de
             CorbaServer EJB1.
   DFHEJ0901 Se ha creado DJar HelloWorldEJB en CorbaServer EJB1.
   DFHEJ5025 Exploración completada para CorbaServer EJB1, 1 DJar creado, 0 DJar
             actualizados.
   DFHEJ5032 Se está publicando automáticamente el contenido de DJar HelloWorldEJB en
             el espacio de nombres.
   DFHEJ5009 Bean HelloWorld publicado en servidor JNDI.
             iiop://nameserver.location.company.com:2809 en ejemplos de ubicación.
   DFHEJ1540 DJar HelloWorldEJB y los beans que contiene ahora son accesibles.
   ```

   donde:

   - **nombre_DJARDIR** es el nombre del directorio (de "recogida") del archivo JAR desplegado del CorbaServer.

- **iiop://nameserver.location.company.com:2809** es el URL y el número de puerto de su servidor de nombres. En este ejemplo, se utiliza un COS Naming Server.

Configuración del cliente:

El código fuente de la aplicación cliente se encuentra en el archivo `HelloWorldCLI.jar`.

Acerca de esta tarea

En z/OS UNIX System Services, debe hacer lo siguientes:

1. Copie el script `runEJBIVP` en un directorio de trabajo. El script `runEJBIVP` original está ubicado, junto con la aplicación de ejemplo de IVP, en el siguiente directorio:

   ```
   /usr/lpp/cicsts/cicsts42/samples/ejb/helloworld
   ```

   donde `cicsts42` es el directorio de instalación para archivos de CICS en z/OS UNIX.
2. Edite la copia del script de `runEJBIVP` como sigue. Esto es necesario para que el cliente pueda localizar el enterprise bean publicado en el espacio de nombres de JNDI. (Un cliente típico no tendrá acceso al perfil de JVM de CICS).
   a. Modifique la variable JAVA_HOME para el directorio de instalación de IBM SDK 6.0.1, como se indica en los comentarios del script. La línea que hay que cambiar es:

      ```
      JAVA_HOME=/usr/lpp/<directorio de instalación de Java SDK java>/J6.0.1_64
      ```

   b. Modifique la variable CICS_HOME a su directorio de instalación para archivos de CICS en z/OS UNIX, como indican los comentarios del script. La línea que hay que cambiar es:

      ```
      CICS_HOME=/usr/lpp/cicsts/<directorio de instalación de CICS>
      ```

   c. Modifique la variable JNDI_PROVIDER_URL al URL y al número de puerto del servidor de nombres, como se indica en los comentarios del script. La línea que hay que cambiar es:

      ```
      JNDI_PROVIDER_URL=iiop://nameserver.location.company.com:2809
      ```

      La línea anterior asume que se está utilizando un servidor de nombres COS, como `tnameserv`, el CORBA Object Services Naming Directory Server ligero incluido con Java 1.3 y posteriores, y que este está configurado para escuchar el puerto 2809.

      Si, por ejemplo, utiliza un servidor de nombres COS configurado para escuchar el puerto 900, puede especificar:

      ```
      JNDI_PROVIDER_URL=iiop://nameserver.location.company.com:900
      ```

      Si utiliza el servidor de nombres `tnameserv`, configurado para escuchar el puerto 2809, en una estación de trabajo llamada `myworkstation.acme.com`, debería especificar:

      ```
      JNDI_PROVIDER_URL=iiop://myworkstation.acme.com:2809
      ```

      Para lanzar el programa `tnameserv`, escriba el siguiente mandato en el indicador de mandatos de la estación de trabajo:

      ```
      tnameserv -ORBInitialPort 2809
      ```

      Si utiliza el CORBA Object Services Naming Directory Server incluido con WebSphere Application Server versión 5 o posteriores, configurado para escuchar el puerto 2809, debería especificar:

```
JNDI_PROVIDER_URL=iiop://nameserver.location.company.com:2809/domain/legacyRoot
```

Si utiliza un servidor de nombres LDAP, el protocolo debería ser `ldap` en lugar de `iiop`; el número de puerto debería ser 389. Por ejemplo:

```
JNDI_PROVIDER_URL=ldap://nameserver.location.company.com:389
```

d. Si utiliza un servidor de nombres LDAP, modifique las variables LDAP_CONTAINERDN y LDAP_NODEROOTDN, como se indica en los comentarios del script.

   Si utiliza un servidor de nombres COS, estas propiedades se ignoran.

e. Si es necesario, modifique la variable INITIAL_CONTEXT_FACTORY, como indican los comentarios del script. Normalmente, puede dejar esta propiedad en el valor predeterminado. Sin embargo, no es posible acceder a algunos proveedores de servicios JNDI utilizando la fábrica de contexto inicial predeterminada. Por ejemplo, si utiliza WebSphere Application Server como proveedor de JNDI, debe definir esta variable como `com.ibm.websphere.naming.WsnInitialContextFactory`.

f. Si ha configurado su CorbaServer y ha instalado el IVP del modo sugerido, las variables CORBASERVER_JNDI_PREFIX y BEAN_NAME ya estarán definidas con los valores correctos. Consulte los comentarios en el script.

Ejecución del programa de verificación de instalación (IVP) de EJB

Para ejecutar el programa de verificación de instalación de EJB, es necesario llevar a cabo los siguientes pasos.

Acerca de esta tarea

Procedimiento

1. Compruebe que el servidor de nombres se está ejecutando.
   a. Para lanzar `tnameserv` en el host local, escriba el siguiente mandato en el indicador de mandatos de z/OS UNIX System Services o de Windows:

      ```
      tnameserv -ORBInitialPort 2809
      ```

      Esto hace que `tnameserv` escuche las conexiones en el puerto TCP/IP 2809.
2. Ejecute el programa cliente IVP desde el directorio de trabajo de z/OS UNIX System Services escribiendo `./runEJBIVP`. En el terminal de z/OS UNIX System Services, debe ver mensajes similares a los siguientes:

```
CICS EJB IVP: Querying the Java SDK level
java version "1.6.0"
Java(TM) SE Runtime Environment (build pmz6460_26-20110218_01)
IBM J9 VM (build 2.6, JRE 1.6.0 z/OS s390x-64 20110217_75924 (JIT enabled, AOT enabled)
J9VM - R26_Java626_GA_20110217_1713_B75924
JIT - r11_20110215_18645
GC - R26_Java626_GA_20110217_1713_B75924
J9CL - 20110217_75924)
JCL - 20110207_01
CICS EJB IVP: Starting the EJB client program
HelloWorld client program started
Performing JNDI lookup using CosNaming
Testing the following location: samples/HelloWorld
Located home interface for HelloWorld bean
You said: Hello from CICS EJB IVP client
HelloWorld client program ended
CICS EJB IVP: Completed successfully
```

Nota:

a. En este ejemplo, se ha utilizado un COS Naming Server. Si se utiliza un servidor de nombres LDAP, se producen mensajes similares.
b. Si se obtiene una `javax.naming.CommunicationException`, tal vez se deba a que el nombre de host de MVS sea incorrecto en el archivo `tcpip.data`. Tal vez pueda arreglar el problema añadiendo una entrada para el sistema MVS en su archivo `/etc/hosts`. Para obtener orientación, consulte los manuales de MVS.

En el archivo de salida estándar de JVM (stdout), debería ver el siguiente mensaje:

```
CICS EJB hello world sample called with string: Hello from CICS EJB IVP client
```

3. Tras ejecutar el IVP, debe realizar los siguientes pasos.
   a. Descarte las definiciones de recurso que creara en `mygroup`.
   b. Si desactivó la seguridad basada en roles de EJB antes de ejecutar el IVP, vuelva a activarla. Para ello, reinicie CICS con el parámetro de inicialización del sistema **XEJB** definido como 'YES'.

Ejecución de las aplicaciones de ejemplo de EJB

Las aplicaciones de ejemplo de EJB requieren un servidor EJB de CICS.

Importante

Debe configurar CICS, como se describe en “Configuración de un servidor EJB” en la página 258, antes de intentar instalar las aplicaciones de ejemplo.

CICS proporciona las siguientes aplicaciones de ejemplo de EJB:

El programa de verificación de instalación (IVP) de EJB
Una aplicación simple que se puede utilizar para probar el entorno de EJB de CICS y el servidor de nombres. No es necesario un servidor web. Consulte “Utilización del procedimiento de verificación de instalación (IVP) de EJB” en la página 276.

La aplicación de ejemplo "Hello World" de EJB
Una aplicación simple que se puede utilizar para probar el entorno de EJB, incluidos CICS, el servidor de nombres y el servidor web. Consulte “La aplicación de ejemplo “Hello World” de EJB”.

La aplicación de ejemplo EJB Bank Account
Una aplicación más compleja que demuestra cómo se pueden utilizar enterprise beans para hacer que información existente controlada por CICS esté disponible para los usuarios web. Consulte “La aplicación de ejemplo EJB Bank Account” en la página 290.

La aplicación de ejemplo “Hello World” de EJB

“Hello World” es una aplicación simple que se puede utilizar para probar el entorno de EJB, incluidos CICS, el servidor de nombres y el servidor web.

Qué hace la aplicación de ejemplo “Hello World” de EJB:

La aplicación de ejemplo solicita una entrada, añade la entrada a un mensaje estándar y visualiza la serie resultante.

La aplicación de ejemplo consta de:
- Un formulario HTML.
- Un servlet Java, más JavaServer Pages (JSP), que se ejecutan en un servidor de aplicaciones web compatible con J2EE.

- Un enterprise bean que se ejecuta en un servidor EJB de CICS.

La aplicación de ejemplo funciona de esta manera:

1. El usuario lanza la aplicación desde un navegador web. Se visualiza un formulario.
2. El formulario pide al usuario que escriba una frase. Cuando el usuario pulsa el botón SUBMIT, se invoca el servlet.
3. El servlet:
   a. Busca una referencia al enterprise bean en el espacio de nombres JNDI.
   b. Crea una nueva instancia remota del enterprise bean en CICS.
   c. Invoca un método en la instancia de bean y pasa como entrada la frase escrita por el usuario.
4. El enterprise bean añade la frase del usuario a la serie "You said" (Usted dijo) y devuelve el resultado al servlet.
5. El servlet utiliza una JavaServer Page para visualizar el resultado en el navegador web del usuario.

La Figura 18 muestra los componentes de la aplicación de ejemplo.Los elementos principales de la aplicación de ejemplo son un servlet Java y un enterprise bean. En este ejemplo, el servlet se ejecuta en un servidor de aplicaciones web de un servidor Windows; se utiliza un COS Naming Server. Pueden usarse otras configuraciones. Por ejemplo, se podría haber utilizado un servidor de nombres LDAP, o el COS Naming Server podría no alojarse en el mismo servidor de aplicaciones que el servlet.

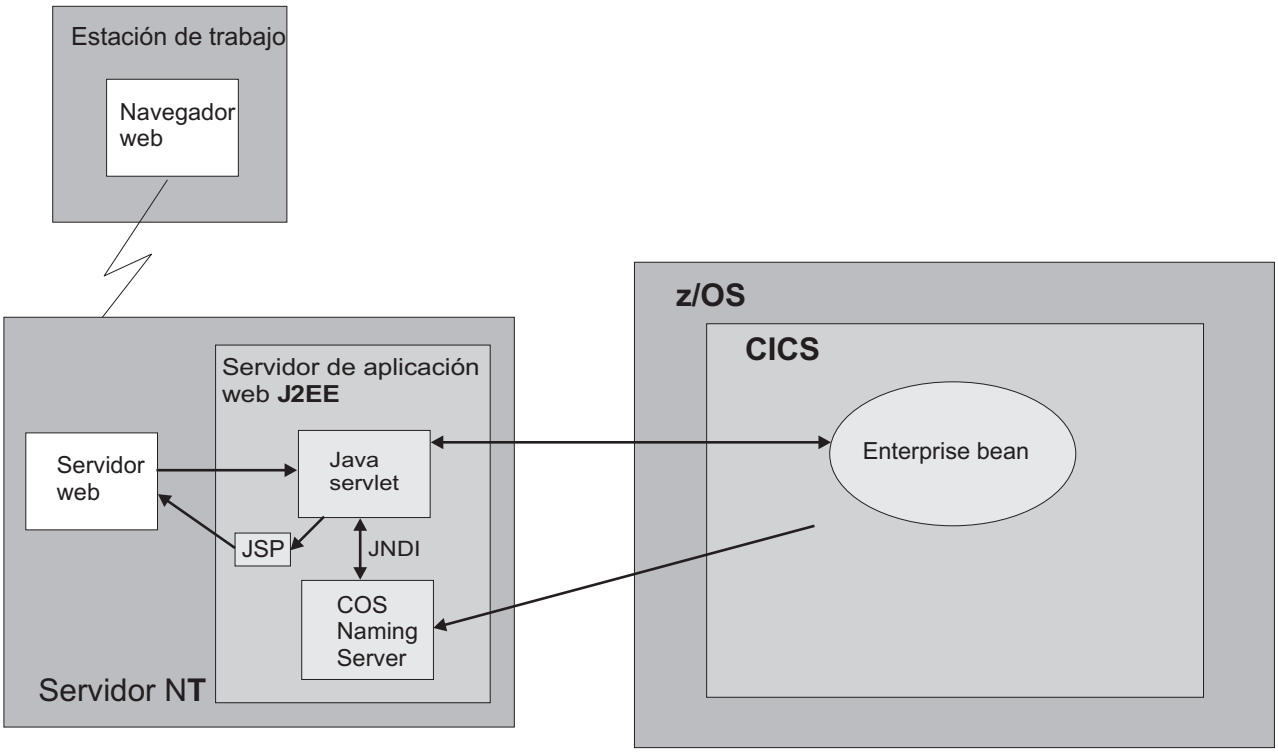


Figura 18. Visión general de la aplicación de ejemplo "Hello World" de EJB

Requisitos previos para la aplicación de ejemplo "Hello World" de EJB:

Necesita estos recursos para ejecutar la aplicación de ejemplo "Hello World" de EJB.

- Un servidor EJB de CICS. La manera de configurar uno se describe en "Configuración de un servidor EJB" en la página 258.
- Un servidor de aplicaciones web que soporte J2EE versión 1.2.1 o posterior. Si se utiliza WebSphere Application Server, tenga en cuenta que el ejemplo requiere WebSphere Application Server versión 4 o posterior.
- Un servidor de nombres que soporte la Java Naming and Directory Interface (JNDI) versión 1.2 ó posterior. La manera de configurar uno se describe en "Acciones necesarias en z/OS o Windows NT" en la página 259.

Componentes suministrados del programa de ejemplo "Hello World" de EJB:

Estos archivos se suministran con el programa de ejemplo "Hello World" de EJB.

Tabla 16. Componentes suministrados del programa de ejemplo "Hello World" de EJB

Nombre de archivo	Tipo	Ubicación predeterminada	Comentarios
`CICSHelloWorld.ear`	Archivo EAR	Directorio de ejemplos de z/OS UNIX: vea la Nota.	Los componentes web de la aplicación de ejemplo: clases y archivos de origen del servlet Java; HTML y JSP.
DFH$EJB	Grupo de definición de recurso	CSD	Contiene las definiciones de recurso de CICS que necesita la aplicación de ejemplo.
`HelloWorldCLI.jar`	Archivo JAR	Directorio de ejemplos de z/OS UNIX: vea la Nota.	Apéndices EJB del cliente que necesita el servlet.
`HelloWorldEJB.jar`	Archivo JAR desplegado	Directorio de ejemplos de z/OS UNIX: vea la Nota.	Clases Java, archivos de origen, descriptor de despliegue, más las clases de soporte para el enterprise bean de CICS. No es necesario desempaquetarlos si no se quiere modificar el código fuente.
`readme.txt`	Archivo de texto	Directorio de ejemplos de z/OS UNIX: vea la Nota.	Contiene: 1. Instrucciones paso a paso para instalar los componentes web de la aplicación de ejemplo "Hello World" de EJB en WebSphere Application Server. 2. Consejos, sugerencias e información de depuración.
Nota: El directorio de ejemplos predeterminado de z/OS UNIX es `/usr/lpp/cicsts/cicsts42/samples/ejb/helloworld` donde `/usr/lpp/cicsts/cicsts42` es el directorio de instalación para archivos de CICS en z/OS UNIX.			

Instalación de la aplicación de ejemplo "Hello World" de EJB:

Debe configurar estos recursos para instalar la aplicación de ejemplo "Hello World" de EJB.

1. z/OS UNIX. Si ha ejecutado anteriormente el IVP de EJB, ya habrá realizado esta acción.
2. CICS. Si ha ejecutado anteriormente el IVP de EJB, ya habrá realizado estas acciones.
3. El servidor de aplicaciones web.

Configuración de z/OS UNIX para la aplicación de ejemplo EJB "Hello World":

Si es necesario, en z/OS UNIX copie el archivo JAR desplegado `HelloWorldEJB.jar` del directorio de ejemplos de EJB al directorio del archivo JAR desplegado de su CorbaServer (directorio de recogida).

Nota:

1. Esto tiene que hacerlo únicamente si no ha instalado ya el archivo JAR desplegado `HelloWorldEJB.jar` mientras se ejecutaba el IVP de EJB.
2. El directorio del archivo JAR desplegado es el directorio que creara en "Antes de ejecutar el procedimiento de verificación de instalación (IVP) de EJB" en la página 258 y que especificara en la opción DJARDIR de la definición CORBASERVER.
3. El directorio de ejemplos es: `/usr/lpp/cicsts/cicsts42/samples/ejb/helloworld`, donde `/usr/lpp/cicsts/cicsts42` es el directorio de instalación para archivos de CICS en z/OS UNIX.
4. Recuerde que los nombres de z/OS UNIX distinguen entre mayúsculas y minúsculas.
5. El archivo `HelloWorldEJB.jar` contiene tanto el código fuente como el código ejecutable para el enterprise bean.

Configuración de CICS:
Acerca de esta tarea

1. Si la seguridad basada en rol de EJB está activa en su región CICS, debe desactivarla antes de ejecutar la aplicación de ejemplo "Hello World" de EJB. Es decir, si ambos parámetros de inicialización del sistema SEC y XEJB actualmente especifican 'YES', debe definir XEJB como 'NO' y reiniciar CICS.
2. El grupo de ejemplo proporcionado por CICS, DFH$EJB, contiene las definiciones TCPIPSERVICE y CORBASERVER adecuadas para ejecutar la aplicación de ejemplo "HelloWorld" de EJB. Debe modificar algunos de los atributos de estas definiciones de recurso para adecuarlas a su propio entorno e instalar las definiciones cambiadas en CICS. Ya debería de haber hecho esto, como parte de la tarea de configurar el servidor EJB. Si no lo ha hecho, siga las instrucciones paso a paso que se muestran en "Acciones necesarias en CICS" en la página 260.

   Nota: El grupo DFH$EJB no contiene ninguna definición REQUESTMODEL, ya que no es necesario instalar ninguna. El ejemplo utiliza el ID de transacción predeterminado, CIRP.

   a. Si es necesario, emita un mandato `CEMT PERFORM CORBASERVER(EJB1) SCAN`. (Tiene que hacer esto únicamente si no ha instalado ya el archivo JAR desplegado `HelloWorldEJB.jar` mientras se ejecutaba el IVP de EJB). CICS:
      1) Explora el directorio de recogida.
      2) Copia el archivo JAR desplegado `HelloWorldEJB.jar` que encuentra en el directorio de recogida en su directorio de estantería.
      3) Crea e instala dinámicamente una definición DJAR para `HelloWorldEJB.jar`.
      4) Debido a que la definición CORBASERVER especifica AUTOPUBLISH(YES), publica el enterprise bean contenido en `HelloWorldEJB.jar` en el espacio de nombres de JNDI.
3. Si no lo ha hecho ya, defina el estado de TCPIPSERVICE como OPEN:

   ```
   CEMT SET TCPIPSERVICE(EJBTCP1) OPEN
   ```

Si emitió el mandato CEMT PERFORM CORBASERVER(EJB1) SCAN, en la CICS Console, debe ver, entre otros, mensajes similares a los siguientes:

```
DFHEJ5024 Comienzo de exploración para CorbaServer EJB1, el directorio que
           se explora es          nombre_DJARDIR.
DFHEJ5030 Se está creando el nuevo DJar HelloWorldEJB durante una exploración de
          CorbaServer EJB1.
DFHEJ0901 Se ha creado DJar HelloWorldEJB en CorbaServer EJB1.
DFHEJ5025 Exploración completada para CorbaServer EJB1, 1 DJar creado, 0 DJar
          actualizados.
DFHEJ5032 Se está publicando automáticamente el contenido de DJar HelloWorldEJB en
          el espacio de nombres.
DFHEJ5009 Bean HelloWorld publicado en servidor JNDI.
          iiop://nameserver.location.company.com:900 en ejemplos de ubicación.
DFHEJ1540 DJar HelloWorldEJB y los beans que contiene ahora son accesibles.
```

donde:

- **nombre_DJARDIR** es el nombre del directorio ("de recogida") del archivo JAR desplegado del CorbaServer.
- **iiop://nameserver.location.company.com:900** es el URL y el número de puerto de su servidor de nombres. En este ejemplo, se utiliza un COS Naming Server.

Configuración del servidor de aplicaciones web:

En el servidor de aplicaciones web debe instalar los componentes web de la aplicación de ejemplo “Hello World” de EJB.

Acerca de esta tarea

Del directorio de ejemplos EJB de z/OS UNIX, necesita:

- `CICSHelloWorld.ear`. Un archivo de archivador empresarial (EAR) J2EE que contiene los componentes web de la aplicación de ejemplo y del código fuente del servlet y de los JSP.
- `readme.txt`. Un archivo de texto que contiene:
  1. Instrucciones paso a paso para instalar los componentes web de la aplicación de ejemplo en WebSphere Application Server.
  2. Consejos, sugerencias e información de depuración.

Nota: El directorio de ejemplos predeterminado es:

```
/usr/lpp/cicsts/cicsts42/samples/ejb/helloworld
```

donde `/usr/lpp/cicsts/cicsts42` es el directorio de instalación para archivos de CICS en z/OS UNIX.

Importante: El resto de esta sección contiene instrucciones para instalar los componentes web del ejemplo en un servidor de aplicaciones web compatible con J2EE (que puede ser WebSphere o no). Es adecuado para usuarios experimentados. Si el servidor de aplicaciones web es WebSphere Application Server versión 4 o posterior *y* usted es un usuario novato de ese producto, le recomendamos que siga en su lugar las instrucciones detalladas específicas de WebSphere que aparecen en el archivo `readme.txt`.

1. Instale los componentes web de la aplicación de ejemplo “Hello World” de EJB (que están en `CICSHelloWorld.ear`) en el servidor de aplicaciones web J2EE siguiendo las directrices del proveedor para la instalación de aplicaciones. En WebSphere Application Sever, por ejemplo, esto implica utilizar la consola de administración para:

a. Instalar una nueva aplicación.
b. Generar el plugin de servidor web actualizado.
c. Guardar la configuración.

Nota: `CICSHelloWorld.ear` incluye una configuración predeterminada para la aplicación de ejemplo "Hello World" de EJB. Para ejecutar la aplicación de ejemplo no es necesario editar ni añadir ninguna información de configuración.

2. Lance la aplicación utilizando el procedimiento estándar del servidor de aplicaciones web.

Prueba de la aplicación de ejemplo "Hello World" de EJB:

Debe realizar estos pasos para probar la aplicación.

Acerca de esta tarea

1. Asegúrese de que se están ejecutando todos los elementos siguientes:
   - El servidor web.
   - El servidor de aplicaciones web y la aplicación de ejemplo.
   - El servidor de nombres.
   - La región CICS
2. Inicie un navegador web y apunte el mismo al URL del servidor web, seguido por "cicshello". Por ejemplo:
   ```
   http://myServer.ibm.com/cicshello
   ```

   Aparece la pantalla inicial que se muestra en la Figura 19 en la página 287.

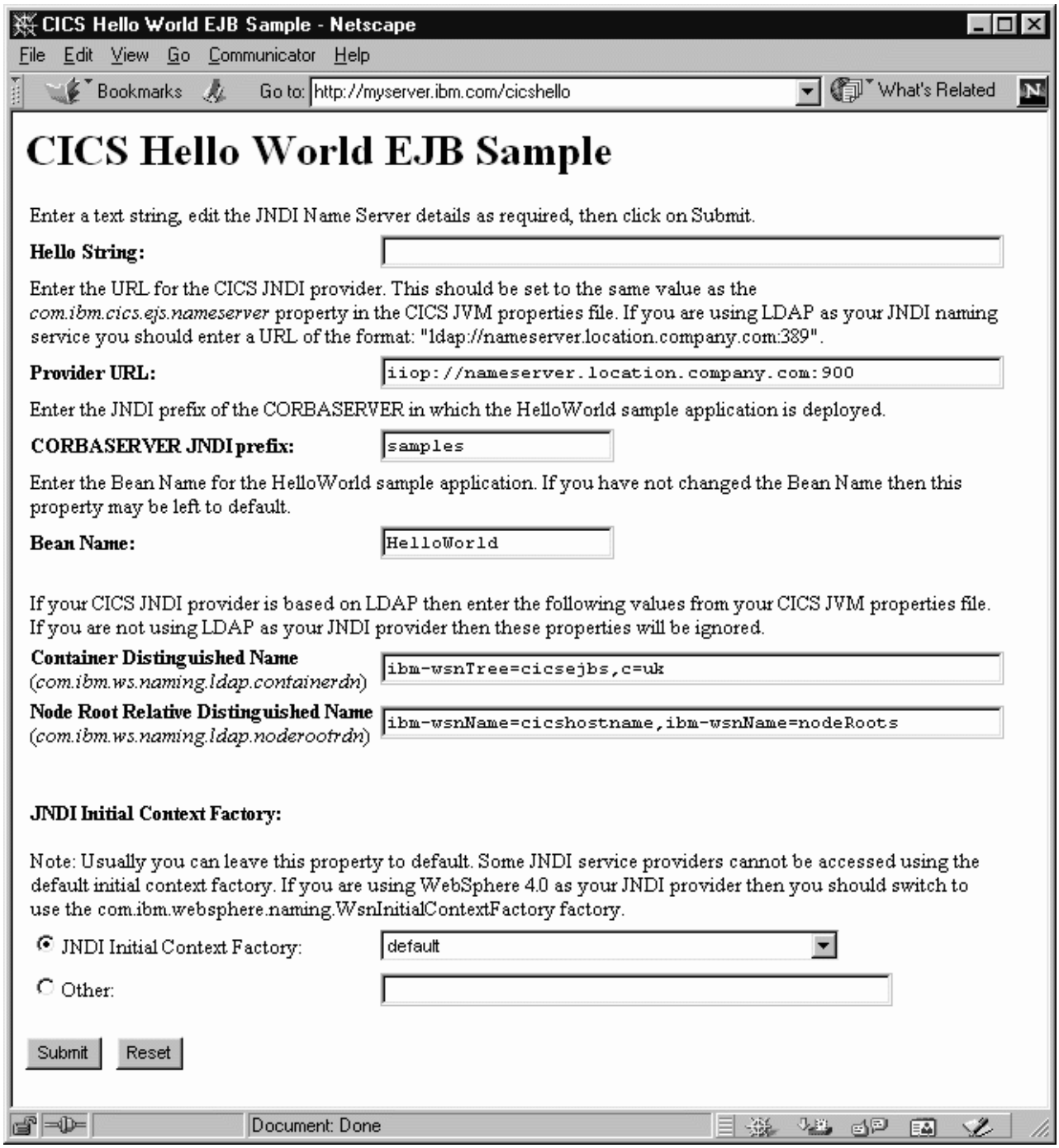


Figura 19. Pantalla inicial de la aplicación de ejemplo "Hello World" de EJB

3. Escriba una frase en el campo `Hello String:`.
4. Compruebe que los campos `Provider URL:`, `CORBASERVER JNDI prefix:`, `Bean Name:`, `Container Distinguished Name:`, `Node Root Relative Distinguished Name:` y `JNDI Initial Context Factory:` contienen valores que son válidos para la instalación. Si no es así, sobrescríbalos del siguiente modo:

   `Provider URL:`
   Escriba el URL y el número de puerto del servidor de nombres en el que se

publica el enterprise bean. (Estos los especifica la propiedad **-Dcom.ibm.cics.ejs.nameserver** del archivo de propiedades de JVM). Por ejemplo:

- Si utiliza un servidor de nombres LDAP con un URL de `myldapns.ibm.com` y un número de puerto de 389, especifique `"ldap://myldapns.ibm.com:389"`.
- Si utiliza un COS Naming Server estándar con un URL de `mycosns.ibm.com` y un número de puerto de 900, especifique `"iiop://mycosns.ibm.com:900"`.
- Si utiliza el CORBA Object Services Naming Directory Server incluido con WebSphere Application Server versión 5 o posteriores, con un URL de `mycosns.ibm.com` y un número de puerto de 2809, especifique:

  ```
  -Dcom.ibm.cics.ejs.nameserver=iiop://mycosns.ibm.com:2809/domain/legacyRoot
  ```

Para obtener información detallada sobre cómo especificar la ubicación del servidor de nombres, consulte la descripción de la propiedad **-Dcom.ibm.cics.ejs.nameserver** en "Propiedades del sistema de la JVM" en la página 118.

CORBASERVER JNDI prefix:
Escriba el prefijo JNDI del CorbaServer. Si utiliza la definición de CORBASERVER incluida en DFH$EJB, no tiene que cambiar el valor predeterminado de "`samples`".

Bean name:
Escriba el nombre del enterprise bean utilizado por la aplicación de ejemplo, como se define en el descriptor de despliegue que hay en el archivo `HelloWorldEJB.jar` incluido. *A no ser que haya renombrado el bean, no es necesario cambiar el valor predeterminado de "HelloWorld".*

Container Distinguished Name:
Si utiliza un servidor de nombres LDAP, escriba el nombre distinguido de la raíz del espacio de nombres del sistema LDAP, como se lo proporcione su administrador de LDAP. (El nombre distinguido de la raíz del espacio de nombres del sistema LDAP se especifica mediante la propiedad **-Dcom.ibm.ws.naming.ldap.containerdn** en el archivo de propiedades de JVM). *Si utiliza un servidor de nombres COS, el valor de este campo se ignora.*

Node Root Relative Distinguished Name:
Si utiliza un servidor de nombres LDAP, escriba el nombre distinguido de la raíz del nodo LDAP, como se lo proporcione su administrador de LDAP. (El nombre distinguido del nombre la raíz del nodo de LDAP lo especifica la propiedad **-Dcom.ibm.ws.naming.ldap.noderootrdn** en el archivo de propiedades de JVM). *Si utiliza un COS Naming Server, el valor de este campo se ignora.*

JNDI Initial Context Factory:
Seleccione la fábrica de contexto inicial de JNDI adecuada de la lista desplegable. Si el servidor de aplicaciones web es WebSphere, la fábrica que se utilizará depende de:

- La versión de WebSphere que utilice.
- La ubicación WebSphere: es decir, si está en una plataforma distribuida como Windows NT o en una plataforma de host como z/OS
- El tipo de servidor de nombres que utiliza: COS Naming Server o LDAP

Tabla 17 en la página 289 muestra la fábrica de contexto inicial correcta para especificar, si el servidor de aplicaciones web es WebSphere.

Tabla 17. Definición de la fábrica de contexto inicial con arreglo a la versión y a la ubicación de WebSphere y al tipo del servidor de nombres

Versión de WebSphere	Ubicación del servidor de aplicaciones web.	Tipo de servidor de nombres.	Fábrica de contexto inicial que utilizar.
3.5	Distribuido	COS	`com.ibm.ejs.ns.jndi.CNInitialContextFactory`
3.5	Distribuido	LDAP	`com.ibm.jndi.LDAPCtxFactory`
3.5	z/OS	COS	`com.sun.jndi.cosnaming.CNCtxFactory`
3.5	z/OS	LDAP	`com.sun.jndi.ldap.LdapCtxFactory`
4 o posterior	Distribuido	COS o LDAP	`com.ibm.websphere.naming.WsnInitialContextFactory`
4 o posterior	z/OS	COS	`com.sun.jndi.cosnaming.CNCtxFactory`
4 o posterior	z/OS	LDAP	`com.sun.jndi.ldap.LdapCtxFactory`

Si el servidor de aplicaciones web no es WebSphere, elija el valor adecuado de la lista desplegable:

Nota: La lista desplegable contiene varias clases de fábrica de contexto inicial, más un elemento de lista "default" (predeterminado). La aplicación de ejemplo asigna el valor del elemento de lista predeterminado del siguiente modo:

a. Si se encuentra la clase `com.ibm.websphere.naming.WsnInitialContextFactory` en la vía de acceso de clases Java, el ejemplo la convierte en el elemento predeterminado. Esta clase es un "derivador" que recorta tanto `com.ibm.ejs.ns.jndi.CNInitialContextFactory` como `com.ibm.jndi.LDAPCtxFactory`. La aplicación de ejemplo determina la clase base correcta examinando el tipo de servidor de nombres que ha especificado en el campo **Provider URL**: si el protocolo especificado es "iiop", la aplicación de ejemplo utiliza `com.ibm.ejs.ns.jndi.CNInitialContextFactory`; si es "ldap", utiliza `com.ibm.jndi.LDAPCtxFactory`.

b. Si la clase `com.ibm.websphere.naming.WsnInitialContextFactory` *no* se encuentra en la vía de acceso de clases Java, la aplicación de ejemplo determina la clase base correcta examinando el tipo de servidor de nombres que ha especificado en el campo **Provider URL**: si el protocolo especificado es "iiop", la aplicación de ejemplo utiliza `com.ibm.ejs.ns.jndi.CNInitialContextFactory`; si es "ldap", utiliza `com.ibm.jndi.LDAPCtxFactory`.

Si ninguno de los valores de la lista desplegable es válido para la instalación, seleccione el botón de selección `Other` y escriba el valor correcto en el campo de texto inferior.

5. Pulse el botón SUBMIT. Esto invoca el servlet y ejecuta la aplicación.

   Si la aplicación está configurada correctamente y los valores de entrada son válidos, las JSP de HelloWorldResults visualizan el mensaje "You said *su frase*" en el navegador web (donde *su frase* es la frase que escribió en el paso 3).

   Si la aplicación no está configurada correctamente, o uno o más de los valores de entrada no es válido, las JSP de HelloWorldError visualizan un mensaje de error en el navegador web. El archivo `readme.txt` contiene sugerencias y consejos que pueden ayudarle a depurar una aplicación con anomalías.

La aplicación de ejemplo EJB Bank Account

La aplicación de ejemplo EJB Bank Account demuestra cómo se pueden utilizar enterprise beans y DB2 para hacer que información existente controlada por CICS esté disponible para los usuarios web.

Qué hace la aplicación de ejemplo EJB Bank Account:

La aplicación de ejemplo extrae información del cliente de tablas de datos y la devuelve al usuario.

La aplicación de ejemplo consta de:

- Un formulario HTML.
- Un servlet Java, más JavaServer Pages que se ejecutan en un servidor de aplicaciones web compatible con J2EE.
- Un enterprise bean que se ejecuta en un servidor EJB de CICS.
- Dos tablas de datos DB2 que contienen información del cliente. Una contiene información de cuenta, como el saldo actual, mientras que la otra contiene detalles sobre el nombre y la dirección.
- Dos programas de servidor de CICS escritos en COBOL. El programa DFH0ACTD recupera información de la tabla de datos de cuentas. El programa DFH0CSTD recupera información de la tabla de datos de nombre y dirección.

La aplicación de ejemplo funciona de este modo:

1. El usuario lanza la aplicación desde un navegador Web. Se visualiza un formulario.
2. El formulario solicita un número de cliente al usuario. Cuando el usuario ha especificado un número de cliente y ha pulsado el botón SUBMIT, se invoca el servlet.
3. El servlet:
   a. Busca una referencia al enterprise bean en el espacio de nombres JNDI.
   b. Crea una nueva instancia remota del enterprise bean en CICS.
   c. Invoca un método en la instancia de bean y pasa como entrada el número de cliente escrito por el usuario.
4. El enterprise bean utiliza la interfaz de cliente común (CCI) del CCI Connector for CICS CICS TS para enlazar con los programas de servidor COBOL de CICS y pasa el número de cliente.

   El CCI Connector for CICS TS se describe en "El CCI Connector for CICS TS" en la página 337.
5. Los programas de servidor utilizan el número especificado como la clave para los registros de DB2 correspondientes a este cliente. Recuperan los detalles del cliente de las tablas de datos de DB2 y devuelven el número de cuenta, el saldo y la dirección al enterprise bean.
6. El enterprise bean devuelve los detalles del cliente al servlet, que utiliza una JavaServer Page para visualizarlos en el navegador web del usuario. Si el número de cliente no es válido, el navegador web visualiza una página de errores.

La Figura 20 en la página 291 muestra los componentes de la aplicación de ejemplo.Los elementos principales de la aplicación de ejemplo son un servlet Java, un enterprise bean, dos programas de servidor de CICS y dos tablas de datos DB2. La aplicación de ejemplo extrae detalles del cliente de tablas de datos y lo devuelve al usuario. En este ejemplo, el servlet se ejecuta en un servidor de aplicaciones web de un servidor Windows; se utiliza un servidor de nombres

LDAP. Pueden usarse otras configuraciones. Por ejemplo, podría utilizarse un COS Naming Server.

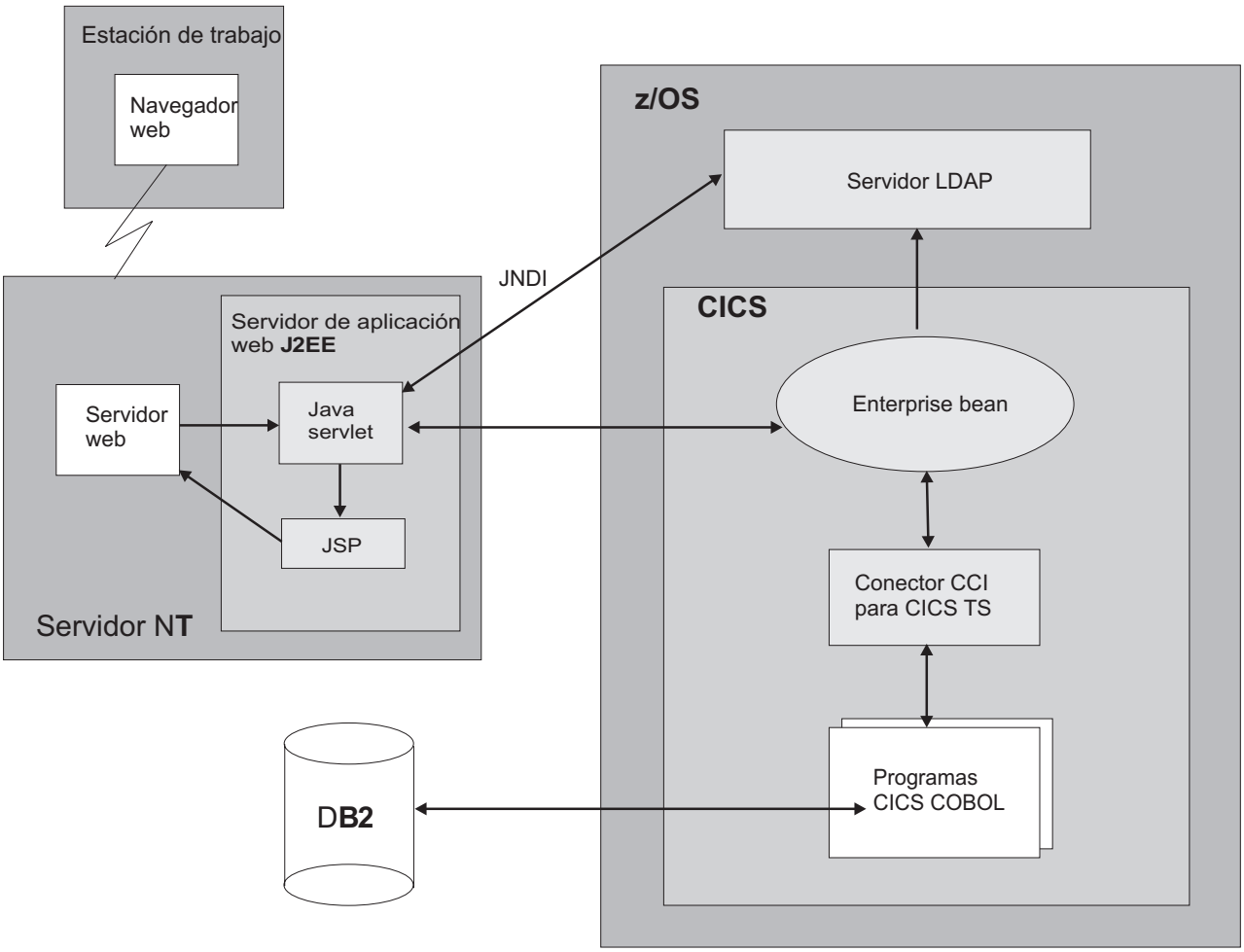


Figura 20. Visión general de la aplicación de ejemplo EJB Bank Account

Requisitos previos de la aplicación de ejemplo EJB Bank Account:

Necesitará estos recursos para ejecutar la aplicación de ejemplo EJB Bank Account.

- Un servidor EJB de CICS. La manera de configurar uno se describe en "Configuración de un servidor EJB" en la página 258.
- DB2 versión 7 o posterior.
- Un servidor de aplicaciones web que soporte J2EE versión 1.2.1 o posterior. Si se utiliza WebSphere Application Server, tenga en cuenta que la aplicación de ejemplo requiere WebSphere Application Server versión 4 ó posterior.
- Un servidor de nombres que soporte JNDI versión 1.2 ó posterior. La manera de configurar uno se describe en "Acciones necesarias en z/OS o Windows NT" en la página 259.

Componentes incluidos de la aplicación de ejemplo EJB Bank Account:

Estos archivos se proporcionan con la aplicación de ejemplo EJB Bank Account.

Tabla 18. Componentes incluidos de la aplicación de ejemplo EJB Bank Account

Nombre de archivo	Tipo	Ubicación predeterminada	Comentarios
DFH$EDB2	Unidad de texto	SDFHSAMP	Sentencias de lenguaje de definición de datos (DDL) de DB2 para definir las tablas de datos de DB2 que utiliza la aplicación de ejemplo y que se llenan de datos.
DFH$ESQL	Unidad de texto	SDFHSAMP	Sentencias del lenguaje de manipulación de datos (DML) de DB2 para enlazar las tablas de datos de DB2 para los programas de servidor COBOL.
DFH$EJB2	Grupo de definición de recurso	CSD	Contiene las definiciones de recurso de CICS que necesita la aplicación de ejemplo.
DFH0ACTD	Código fuente COBOL	SDFHSAMP	Código fuente del programa de servidor DFH0ACTD.
DFH0CSTD	Código fuente COBOL	SDFHSAMP	Código fuente del programa de servidor DFH0CSTD.
DFHEBURM	Programa de ejemplo sustituible por el usuario	SDFHSAMP	Cambia el ID de usuario en el que se ejecuta la aplicación de ejemplo.
`CicsSample.ear`	Archivo EAR	Directorio de ejemplos de z/OS UNIX: vea la Nota.	Los componentes web de la aplicación de ejemplo: clases y archivos de origen del servlet Java; HTML y JSP.
`readme.txt`	Archivo de texto	Directorio de ejemplos de z/OS UNIX: vea la Nota.	Contiene: 1. Contiene instrucciones paso a paso para instalar los componentes web de la aplicación de ejemplo EJB en WebSphere Application Server. 2. Consejos, sugerencias e información de depuración.
`SampleCLI.jar`	Archivo JAR	Directorio de ejemplos de z/OS UNIX: vea la Nota.	Apéndices EJB del cliente que necesita el servlet.
`SampleEJB.jar`	Archivo JAR desplegado	Directorio de ejemplos de z/OS UNIX: vea la Nota.	Clases Java, archivos de origen, descriptor de despliegue, más las clases de soporte para el enterprise bean de CICS. No es necesario desempaquetarlos si no se quiere modificar el código fuente.
Nota: El directorio de ejemplos predeterminado de z/OS UNIX es `/usr/lpp/cicsts/cicsts42/samples/ejb/bankaccount` donde `cicsts42` es el directorio de instalación para archivos de CICS en z/OS UNIX.			

Seguridad de la aplicación de ejemplo EJB Bank Account:

Se recomienda ejecutar la aplicación de ejemplo Bank Account en un entorno seguro. Sin embargo, para simplificar el proceso, al principio puede optar por no hacerlo así.

Si no desea activar el entorno seguro inmediatamente, defina el parámetro de inicialización del sistema XEJB como 'NO' y sáltese el resto de la sección. Para activar el entorno seguro posteriormente, siga las instrucciones que aparecen en el resto de esta sección.

Puede implementar la seguridad para la aplicación de ejemplo de distintas formas. Por ejemplo, puede utilizar cualquiera de las siguientes alternativas:

- Permita a todos los usuarios ejecutar la aplicación de ejemplo con el ID de usuario predeterminado.
- Permita a todos los usuarios ejecutar la aplicación de ejemplo con un ID de usuario especificado por el programa de salida de seguridad para IIOP.
- Utilice un certificado del lado del servidor de SSL para cifrar los datos enviados entre el nivel web y CICS, permitiendo a todos los usuarios ejecutar la aplicación de ejemplos en un transporte seguro con el ID de usuario predeterminado.
- Utilice un certificado del lado del servidor de SSL para cifrar los datos enviados entre el nivel web y CICS, permitiendo a todos los usuarios ejecutar la aplicación de ejemplo en un transporte seguro con un ID de usuario especificado por el programa de salida de seguridad para IIOP.
- Utilice un certificado de cliente de SSL para autenticar automáticamente el servidor de aplicaciones del nivel web ante CICS, permitiendo a todos los usuarios ejecutar la aplicación de ejemplo en un transporte seguro, con un ID de usuario asignado al servidor de aplicaciones del nivel web.
- Utilice autenticación de identidad afirmada para permitir que las aplicaciones cliente del nivel web que se ejecutan en WebSphere Application Server for z/OS propaguen sus ID de usuario existentes a CICS sobre un transporte seguro.

Nota:

1. De forma predeterminada, la aplicación Bank Account no necesita que el usuario se autentique en el nivel web. Puede optar por activar la autenticación en el contenedor web siguiendo las instrucciones del servidor de aplicaciones. Si sí que se autentica en el nivel web, el principio de seguridad no se propaga a CICS, por lo que en términos de seguridad de CICS no tiene efecto. Sin embargo, podría utilizarse autenticación temprana en el nivel web para crear un “dominio de protección”, en el que CICS confíe en que el nivel web no permita a los usuarios no autenticados invocar métodos de negocio en enterprise beans de CICS.
2. Para utilizar cifrado o autenticación SSL, necesita un servidor de aplicaciones web compatible con J2EE y con soporte completo de SSL. Consulte la documentación del proveedor para obtener más detalles.
3. Para obtener más información sobre la autenticación SSL, consulte SSL authentication, en la *Guía de seguridad RACF de CICS*.

Sea cual sea el método de autenticación que elija, tiene que hacer (entre otras cosas) lo siguiente:

1. Brindar información de autorización en el descriptor de despliegue del enterprise bean en CICS. Esta información de autorización consta de:

Un elemento "rol de seguridad".
Identifica una clase de usuario a la que se le permite realizar una acción determinada o utilizar un recurso dado.

Un elemento de "permiso de método".
Identifica métodos específicos del enterprise bean que los miembros del rol de seguridad especificado están autorizados a utilizar.

2. Actualice el gestor de seguridad externa (ESM) de CICS para correlacionar el rol de seguridad especificado con distintos ID de usuario reales. Las siguientes instrucciones paso a paso para implementar la seguridad asumen que el ESM que ha elegido es RACF. Si utiliza un ESM distinto, consulte al proveedor de su ESM para obtener orientación.

Implementación de seguridad basada en roles para la aplicación de ejemplo EJB Bank Account:

Puede implementar seguridad basada en roles para la aplicación de ejemplo Bank Account mediante el Assembly Toolkit (ATK, que es un componente del Application Server Toolkit, ASTK).

Acerca de esta tarea

Esta herramienta se incluye como parte de WebSphere Application Server versión 5.1 y posteriores. Puede utilizar la interfaz gráfica de usuario de ATK para (entre otras cosas) editar el contenido del descriptor de despliegue de un enterprise bean.

Antes de empezar, asegúrese de que ATK está instalado en la estación de trabajo. Una vez instalado, la herramienta se puede iniciar desde un icono que se añade al menú de inicio de Windows.

ATK se utiliza para la primera fase de la implementación de seguridad basada en roles, que implica editar el descriptor de despliegue del enterprise bean. Cuando haya completado esa fase, siga las instrucciones para la segunda fase de la implementación de seguridad basada en roles, que implica configurar otro software.

Fase 1. Utilización de ATK para editar el descriptor de despliegue:

En este punto, para familiarizarse con ATK, puede examinar el contenido del archivo JAR.

1. Copie el archivo `SampleEJB.jar` del directorio de ejemplos de z/OS UNIX a la estación de trabajo. Puede hacerlo utilizando FTP en modalidad binaria o cualquier método de su elección. El directorio de ejemplos de z/OS UNIX es `/usr/lpp/cicsts/cicsts42/samples/ejb/bankaccount`. Para ATK, también necesita realizar el mismo proceso para el archivo `dfjcci.jar`, que está en el directorio `/usr/lpp/cicsts/cicsts42/lib`. No es necesario que edite ese archivo JAR, pero ATK tiene que recrear correctamente el archivo JAR para la aplicación de ejemplo EJB Bank Account tras editar.
2. Importe el archivo JAR a ATK como un proyecto EJB.
   a. Lance ATK y vaya a la perspectiva J2EE seleccionando **Window > Open Perspective > J2EE** (Ventana > Abrir perspectiva > J2EE).
   b. Seleccione la opción **Import** (Importar) desde el menú **File** (Archivo). Seleccione **EJB JAR File** (Archivo JAR EJB) como origen de la importación. Seleccione **Browse** (Examinar) y encuentre el archivo `SampleEJB.jar`. Escriba un nombre adecuado para el proyecto. Seleccione **Next** (Siguiente) y elija

importar todos los enterprise beans, que es el valor predeterminado. Seleccione **Finish** (Finalizar) para crear el proyecto EJB.

c. Cuando se cree el proyecto, debería ver que aparecen algunos errores en la lista de tareas. Para corregir estos errores, tiene que añadir el archivo `dfjcci.jar` a la vía de acceso de creación para el proyecto EJB. En el panel de navegación de la izquierda (utilizando la vista de jerarquía de J2EE), expanda el elemento "EJB Modules" (Módulos EJB) para ver el proyecto EJB. Pulse con el botón derecho del ratón en el nombre del proyecto y seleccione **Properties** (Propiedades). Seleccione **Java Build Path** (Vía de acceso de creación de Java). Vaya al separador **Libraries** (Bibliotecas) y seleccione el botón **Add External JARs** (Añadir JAR externos). Navegue hasta el archivo `dfjcci.jar` y seleccione **Open** (Abrir). Seleccione **OK** (Aceptar). ATK recrea el proyecto EJB y los errores deberían desaparecer.

Para obtener más información sobre el descriptor de despliegue de EJB, consulte “Enterprise beans: el descriptor de despliegue” en la página 237.

3. Añada roles de seguridad al descriptor de despliegue. En ATK, en el panel de navegación de la izquierda (utilizando la vista de jerarquía de J2EE), expanda el elemento "EJB Modules" (Módulos EJB) para ver el proyecto EJB. Haga doble pulsación en el nombre del proyecto para abrirlo. Seleccione el separador **Assembly Descriptor** (Descriptor de ensamblaje) en la parte de abajo del panel. En **Security Roles** (Roles de seguridad), seleccione el botón Add (Añadir) para añadir un nuevo rol de seguridad.

   Si su organización ya ha configurado roles de seguridad para otras aplicaciones, tal vez desee reutilizar un rol existente. Si es así, escriba el nombre del rol que desee utilizar en el campo provisto para ello. Si no hay ningún rol de seguridad existente que desee reutilizar, escriba un nuevo nombre de rol, como “Todos_usuarios”. También puede proporcionar una descripción opcional del rol para que le sirva de recordatorio en el futuro. Seleccione **Finish** (Finalizar) para volver a la ventana principal.

   Nota: Si reutiliza un rol de seguridad existente que ya esté definido para su gestor de seguridad externa (ESM), debe eliminar el elemento `Display Name` (Nombre de visualización) del descriptor de despliegue del archivo JAR. CICS utiliza este elemento para proporcionar un nombre de aplicación que tenga un prefijo para todos los nombres de rol de seguridad al realizar una comprobación de seguridad en tiempo de ejecución, y así proporciona soporte para los roles de seguridad indicados en el nivel de aplicación en lugar de para toda la empresa. En ATK, puede eliminar este elemento seleccionando el separador **Overview** (Visión general) en la parte de abajo del panel. Seleccione el texto en el campo `Display Name` y suprímalo.

4. Ahora defina un permiso de método y asócielo con un rol de seguridad. En ATK, vuelva a seleccionar el separador **Assembly Descriptor**. En **Method Permissions** (Permisos de método), seleccione el botón **Add** (Añadir). El asistente presenta una lista de los roles de seguridad que haya definido. Para la aplicación de ejemplo Bank Account, es apropiado ejecutar todos los métodos en el mismo rol de seguridad. Seleccione el rol de seguridad que desea asociar con el permiso de método y seleccione **Next** (Siguiente). Seleccione el bean CICSSample y luego **Next**. Seleccione el recuadro de CICSSample para seleccionar todos los elementos de método para el bean. Seleccione **Finish** (Finalizar). Vuelve a la pantalla anterior.
5. Guarde el descriptor de despliegue actualizado seleccionando la opción **Save** (Guardar) del menú **File** (Archivo).
6. Exporte el proyecto desde ATK de vuelta a un archivo JAR de su estación de trabajo. Para ello, seleccione la opción **Export** (Exportar) desde el menú **File**

(Archivo). Seleccione **EJB JAR File** (Archivo JAR EJB) como destino de exportación y luego seleccione **Next** (Siguiente). Seleccione el proyecto EJB de la lista desplegable. Seleccione **Browse** (Examinar) y localice el archivo `SampleEJB.jar` que utilizar en el destino. (Esto sobrescribe la versión original del archivo. Tal vez desee conservar una copia de seguridad de la versión original de dicho archivo en la estación de trabajo con un nombre distinto). Seleccione el recuadro de selección para **Export source files** (Exportar archivos de origen) para conservar los archivos de origen con el archivo JAR. Seleccione **Finish** (Finalizar). Salga de ATK.

7. Copie el archivo `SampleEJB.jar` actualizado de vuelta a z/OS UNIX. Puede utilizar FTP en modalidad binaria o su proceso preferido de transferencia de archivos. Guarde el archivo `SampleEJB.jar` en el directorio de recogida de su CorbaServer.

Fase 2. Configuración de otros valores de seguridad:

El ID o los ID de usuario de CICS que decida asociar con el rol de seguridad definido en el descriptor de despliegue del enterprise bean deben elegirse de acuerdo con qué implementación de seguridad se seleccionara al inicio de esta sección.

1. Asegúrese de que los parámetros de inicialización del sistema de CICS SEC y XEJB especifican 'YES'. (Si alguno especifica 'NO', la seguridad de EJB basada en rol se desactiva).
2. Si ha reutilizado un rol de seguridad existente que ya se haya configurado en su instalación, puede saltarse este paso, que sirve para actualizar RACF y asociar el rol de seguridad de EJB con un conjunto de ID de usuario de CICS.

   Nota: Si el ESM no es RACF, debe pedir consejo al proveedor de dicho ESM sobre cómo realizar este paso.

   Por ejemplo:

   - Si quiere permitir que todos los usuarios anónimos ejecuten la aplicación de ejemplo (tanto si se utiliza SSL como no), debe asociar el ID de usuario predeterminado CICSUSER con el rol de seguridad.
   - Si quiere ejecutar la aplicación de ejemplo con uno o varios ID de usuario seleccionados por el programa de salida de seguridad para IIOP (tanto si se utiliza SSL como no), debe asociar dichos ID de usuario con el rol de seguridad.
   - Si desea utilizar certificación de cliente SSL completa, debe asociar el ID de usuario del certificado del servidor de aplicaciones de nivel web con el rol de seguridad.

   Para configurar la correlación necesaria de rol de seguridad de EJB con ID de usuario de CICS:

   a. Ejecute el programa de utilidad generador EJBROLE de RACF en el archivo `SampleEJB.jar` actualizado. (El programa de utilidad generador EJBROLE de RACF es un programa Java que extrae información de rol de seguridad del descriptor de despliegue y genera un programa REXX que define los roles de seguridad para RACF. Para obtener información sobre cómo utilizar el programa de utilidad generador, consulte “Utilización del programa de utilidad generador EJBROLE de RACF” en la página 374.)
   b. Pida a su administrador de RACF que ejecute el programa REXX generado por el programa de utilidad generador EJBROLE de RACF.

3. Si no desea utilizar el programa de salida de seguridad para IIOP para modificar el ID de usuario en el que se ejecuta la aplicación de ejemplo (del ID de usuario predeterminado de CICS a otro ID de usuario que seleccione), puede saltarse este paso.

   CICS proporciona un programa de salida de seguridad de ejemplo, DFHEBURM, que altera el ID de usuario en el que se ejecuta la aplicación de ejemplo Bank Account del ID de usuario predeterminado de CICS a "SAMPLE". Puede utilizar esta versión del programa sustituible por el usuario o modificarla para ajustarla a sus necesidades. Si ya dispone de un programa de salida de seguridad para IIOP personalizado, puede actualizar su versión para que realice una función similar.

   Debe especificar el nombre de su programa de salida de seguridad en la opción URM de la definición TCPIPSERVICE en la que se va a ejecutar la aplicación de ejemplo.

   Para obtener información orientativa sobre el programa de salida de seguridad para IIOP, consulte "Utilización del programa de seguridad IIOP sustituible por el usuario" en la página 417.

   Para obtener información sobre la escritura de un programa de salida de seguridad para IIOP, consulte la *Guía de personalización de CICS*. Asimismo, estudie el código fuente del programa de ejemplo proporcionado, que contiene comentarios y sugerencias.

   Para obtener información sobre la compilación y la instalación de programas sustituibles por el usuario, consulte EAssembling and link-editing user-replaceable programs, en la *Guía de personalización de CICS*.

   Para obtener información sobre la codificación de definiciones de TCPIPSERVICE, consulte la *Guía de definición de recurso de CICS*.
4. Si utiliza cifrado o autenticación SSL, debe hacer lo siguiente:
   - Configure su servidor de aplicaciones web compatible con J2EE para utilizar SSL. Consulte la documentación del servidor web para obtener orientación al respecto.
   - Disponga de un certificado del servidor disponible para su utilización.
   - Modifique las definiciones de los recursos CORBASERVER y TCPIPSERVICE con los que se va a ejecutar la aplicación de ejemplo. Es decir:
     – Si utiliza autenticación SSL del lado de cliente, la opción CLIENTCERT de la definición de CORBASERVER debe especificar el nombre de un TCPIPSERVICE que defina el puerto que utilizar para solicitudes IIOP de entrada con certificación de cliente SSL. Asimismo, el certificado SSL del servidor de aplicaciones web debe:
       - Incluirse en la lista de certificados de confianza de CICS, en RACF.
       - Correlacionarse con un ID de usuario de RACF.
     – Si utiliza autenticación SSL del lado del servidor, la opción SSLUNAUTH de la definición de CORBASERVER debe especificar el nombre de un TCPIPSERVICE que defina el puerto que utilizar para solicitudes IIOP de entrada con SSL pero sin certificación de cliente.

     Para obtener información sobre la codificación de Definiciones del recurso CORBASERVER y Definiciones del recurso TCPIPSERVICE, consulte la *Guía de definición de recurso de CICS*.
   - Si utiliza autenticación de identidad por afirmación para cifrado, autenticación y propagación de identidad, debe realizar lo siguiente:
     – Configure WebSphere Application Server for z/OS para autenticar usuarios.

- Si utiliza WebSphere Application Server for z/OS versión 6.1 o posterior, para habilitar un protocolo de autenticación adecuado especifique la propiedad del sistema **-Dcom.ibm.cics.iiop.CSIv2Enabled=true** en todos los archivos de propiedades de JVM utilizados en la región CICS. (La release 6.1.0.13 o posterior de WebSphere Application Server para z/OS es necesaria para el soporte de esta función.)
- Habilite la certificación de cliente SSL en WebSphere.
- Disponga de un certificado SSL del servidor disponible para su utilización en CICS.
- Incluya el certificado del servidor asociado con WebSphere Application Server en la lista de RACF de certificados de confianza de CICS. Además, es necesario otorgar permiso al ID de usuario asociado con el certificado de RACF para afirmar la identidad de otros usuarios.
- Modifique las definiciones de los recursos CORBASERVER y TCPIPSERVICE con los que se va a ejecutar la aplicación de ejemplo. La opción ASSERTED de la definición de CORBASERVER debe especificar el nombre de un TCPIPSERVICE que defina el puerto que utilizar para solicitudes IIOP de entrada con autenticación de identidad con afirmación.

Instalación de la aplicación de ejemplo EJB Bank Account:

La instalación de la aplicación de ejemplo EJB Bank Account requiere ejecutar acciones en:
1. z/OS (DB2 y CICS)
2. El servidor de aplicaciones web

Configuración de z/OS:

Utilice el siguiente procedimiento para instalar el ejemplo EJB en z/OS.

Acerca de esta tarea

1. Compile y edite el enlace de los programas de servidor COBOL DB2 de CICS mediante los procedimientos normales de su organización. Los miembros DFH0ACTD y DFH0CSTD de la biblioteca SDFHSAMP contienen el código fuente de los programas de servidor.

   Almacene los módulos de carga en una librería de carga de aplicación que esté incluida en la concatenación DD DFHRPL de CICS.
2. Defina las tablas de datos de DB2 utilizadas por la aplicación de ejemplo y llénelas con datos. La unidad de texto DFH$EDB2 contiene las sentencias necesarias de DB2 DDL y los datos provistos.

   Antes de utilizar DFH$EDB2, debe modificar las siguientes líneas para adecuarlas a su sistema:

   ```
   CREATE STOGROUP EBSAMPSG VOLUMES(SYSDA,SYSDB) VCAT DSNxxxxx;
   ```

   Cambie*DSNxxxxx* al nombre de su identificador de catálogo de Integrated Catalog Facility (ICF) de alto nivel para conjuntos de datos VSAM definidos por el usuario.

   Autoridad necesaria: autoridad de DB2 para crear una base de datos, grupo de almacenamiento, espacio de tabla, tablas e índices.
3. Enlace las tablas de DB2 con los programas de servidor COBOL. La unidad de texto DFH$ESQL contiene las sentencias necesarias de DB2 DML.

   Autoridad necesaria: autoridad de DB2 para realizar un enlace (BIND) para esta base de datos.

Nota:

a. Este paso enlaza de forma estática las sentencias SQL de los programas de servidor con DB2, de forma que no deban enlazarse de forma dinámica en tiempo de ejecución, lo que mejora el rendimiento en dicho tiempo de ejecución.

b. Si vuelve a compilar uno de los programas de servidor posteriormente y pretende que este acceda a DB2, cada vez que lo vuelva a compilar utilice los pasos siguientes:

   1) Vuelva a enlazar las tablas de DB2 con los programas de servidor COBOL.

   2) Renovar la copia del programa de servidor en CICS ejecutando el siguiente mandato de CICS en la región CICS:

      ```
      CEMT SET PROG(nombre_programa) NEW
      ```

      Por ejemplo, si cambia el programa DFH0CSTD y lo vuelve a compilar, utilice `CEMT SET PROG(DFH0CSTD) NEW`. (DFH0CSTD se define para la región CICS en el grupo de definición de recurso DFH$EJB2; consulte el paso 5).

4. Otorgue autoridad al ID de usuario de CICS para acceder al plan de DB2 utilizando el procedimiento normal de su organización (por ejemplo, SPUFI). Para obtener información sobre cómo otorgar acceso a un plan de DB2, consulte Controlling users' access to plans en la *Guía de DB2 de CICS*.

5. Defina los programas y las conexiones de DB2 que utiliza la aplicación de ejemplo para CICS. El grupo de ejemplo proporcionado por CICS, DFH$EJB2, contiene definiciones de recurso para el ejemplo "Bank Account" de EJB. Debe modificar algunos de los atributos de estas definiciones de recurso para adecuarlas a su propio entorno. Para ello, utilice la transacción CEDA del programa de utilidad DFHCSDUP.

   a. Copie el grupo de ejemplo a un grupo de su elección. Por ejemplo:

      ```
      CEDA COPY GROUP(DFH$EJB2) TO(mygroup)
      ```

   b. Visualice el grupo `mygroup` y cambie los atributos de las siguientes definiciones tal y como se muestra:

      - En la definición de DB2CONN, modifique el valor de DB2ID al ID del subsistema DB2 en el que creara las tablas de DB2 utilizadas por el ejemplo.
      - Las definiciones de PROGRAM no tienen que modificarse.

   c. Descarte las definiciones que no necesite del grupo `mygroup`.

      Además de las definiciones DB2CONN y PROGRAM, DFH$EJB2 también contiene una definición CORBASERVER y una definición TCPIPSERVICE. Sin embargo, estas solo tienen fines de referencia. Le recomendamos encarecidamente que configure su propio servidor EJB, como se describe en "Configuración de un servidor EJB" en la página 258, *antes* de intentar instalar los programas de ejemplo. Si lo hace, no necesitará las definiciones CORBASERVER y TCPIPSERVICE en DFH$EJB2 porque ya habrá creado las suyas propias en base a las proporcionadas en el grupo de recursos DFH$EJB. Descártelas del grupo `mygroup`.

      Si *sí* que decide utilizar las definiciones CORBASERVER y TCPIPSERVICE en DFH$EJB2, debe modificarlas como se describe en "Acciones necesarias en CICS" en la página 260.

      Si su región CICS utiliza el programa de instalación automática , no necesita las definiciones PROGRAM. Descártelas del grupo `mygroup`.

Nota: No se proporciona ninguna definición REQUESTMODEL, ya que no es necesario instalar ninguna. El ejemplo utiliza el ID de transacción predeterminado, CIRP.

d. Añada el grupo de recursos que contiene las definiciones de recurso modificadas al CSD de CICS y a la lista de grupos iniciales de CICS. Puede utilizar el programa de utilidad de definición de sistema de CICS, DFHCSDUP. Consulte Programa de utilidad de archivo de definición de sistema (DFHCSDUP) en la *Guía de definición de recurso de CICS*.

Autoridad necesaria: autoridad RACF para instalar definiciones de recurso en la región CICS.

6. Si no lo ha hecho al configurar la seguridad, ponga el archivo JAR desplegado `SampleEJB.jar` en el directorio de recogida de su CorbaServer.
7. Asegúrese de que el servidor de nombres se ha iniciado. Si no se ha iniciado CICS, láncelo ahora.
8. Emita el siguiente mandato en la consola de la región CICS:

```
CEMT PERFORM CORBASERVER(nombre_corbaserver) SCAN
```

CICS explora el directorio de recogida. copia el archivo JAR desplegado `SampleEJB.jar` a su directorio de estantería e instala una definición DJAR para él.

Nota: Si tuvo que lanzar CICS en el paso 7, este paso no es necesario, ya que CICS habrá explorado el directorio de recogida en el inicio.

Autoridad necesaria: autoridad RACF para crear un DJAR y actualizar el acceso a CORBASERVER.

9. Publique el enterprise bean en el espacio de nombres de JNDI. Si la definición CORBASERVER especifica AUTOPUBLISH(YES), esto sucede automáticamente al instalar el archivo JAR desplegado `SampleEJB.jar`. Si la definición CORBASERVER especifica AUTOPUBLISH(NO), emita el siguiente mandato en la consola de la región CICS:

```
CEMT PERFORM DJAR(SampleEJB) PUBLISH
```

Autoridad necesaria: autoridad RACF para actualizar un DJAR.

10. Utilice el programa de ejemplo `CICSConnectionFactoryPublish` para crear un objeto ConnectionFactory para que lo utilice el CCI Connector for CICS TS y para publicarlo en el espacio de nombres. Para obtener información sobre cómo utilizar el programa `CICSConnectionFactoryPublish`, consulte "Uso de los programas de utilidad de ejemplo para gestionar y adquirir una fábrica de conexiones" en la página 347.
11. Asegúrese de que el estado de conexión de DB2 es CONNECTED (conectado) emitiendo el siguiente mandato en la consola del sistema CICS:

```
CEMT SET DB2CONN CONNECTED
```

Configuración del servidor de aplicaciones web:

En el servidor de aplicaciones web debe instalar los componentes web de la aplicación de ejemplo EJB Bank Account.

Acerca de esta tarea

Del directorio de ejemplos EJB de z/OS UNIX, necesita:

- `CicsSample.ear`. Un archivo de archivador empresarial (EAR) J2EE que contiene los componentes web del ejemplo.
- `readme.txt`. Un archivo de texto que contiene:

1. Instrucciones paso a paso para instalar los componentes web de la aplicación de ejemplo en WebSphere Application Server.
2. Consejos, sugerencias e información de depuración.

Nota: El directorio de ejemplos predeterminado es:
`/usr/lpp/cicsts/cicsts42/samples/ejb/bankaccount`

donde `/usr/lpp/cicsts/cicsts42` es el directorio de instalación para archivos de CICS en z/OS UNIX.

Importante: El resto de esta sección contiene instrucciones para instalar los componentes web de la aplicación de ejemplo en un servidor de aplicaciones web compatible con J2EE (que puede ser WebSphere o no). Resulta adecuado para usuarios experimentados. Si el servidor de aplicaciones web es WebSphere Application Server versión 4 o posterior *y* usted es un usuario novato de ese producto, le recomendamos que siga en su lugar las instrucciones detalladas específicas de WebSphere que aparecen en el archivo `readme.txt`.

Procedimiento

1. Instale los componentes web de la aplicación de ejemplo EJB Bank Account (que están en `CicsSample.ear`) en el servidor de aplicaciones web J2EE siguiendo las directrices del proveedor para instalación de aplicaciones. En WebSphere Application Sever, por ejemplo, esto implica utilizar la consola de administración para:
   a. Instalar una nueva aplicación.
   b. Generar el plugin de servidor web actualizado.
   c. Guardar la configuración.

   Nota: `CicsSample.ear` incluye una configuración predeterminada para la aplicación de ejemplo EJB Bank Account. Para ejecutar la aplicación de ejemplo no es necesario editar ni añadir ninguna información de configuración.
2. Lance la aplicación utilizando el procedimiento estándar del servidor de aplicaciones web.

Resultados

Prueba de la aplicación de ejemplo EJB Bank Account:

Debe realizar estos pasos para probar la aplicación.

Acerca de esta tarea

1. Asegúrese de que se están ejecutando todos los elementos siguientes:
   - El servidor web.
   - El servidor de aplicaciones web y la aplicación de ejemplo.
   - El servidor de nombres.
   - La región CICS
   - El subsistema DB2
2. Inicie un navegador web y apunte con el mismo al URL del servidor web, seguido por "cicssample". Por ejemplo:
   `http://myServer.ibm.com/cicssample`

   Aparece la pantalla inicial que se muestra en la Figura 21 en la página 302.

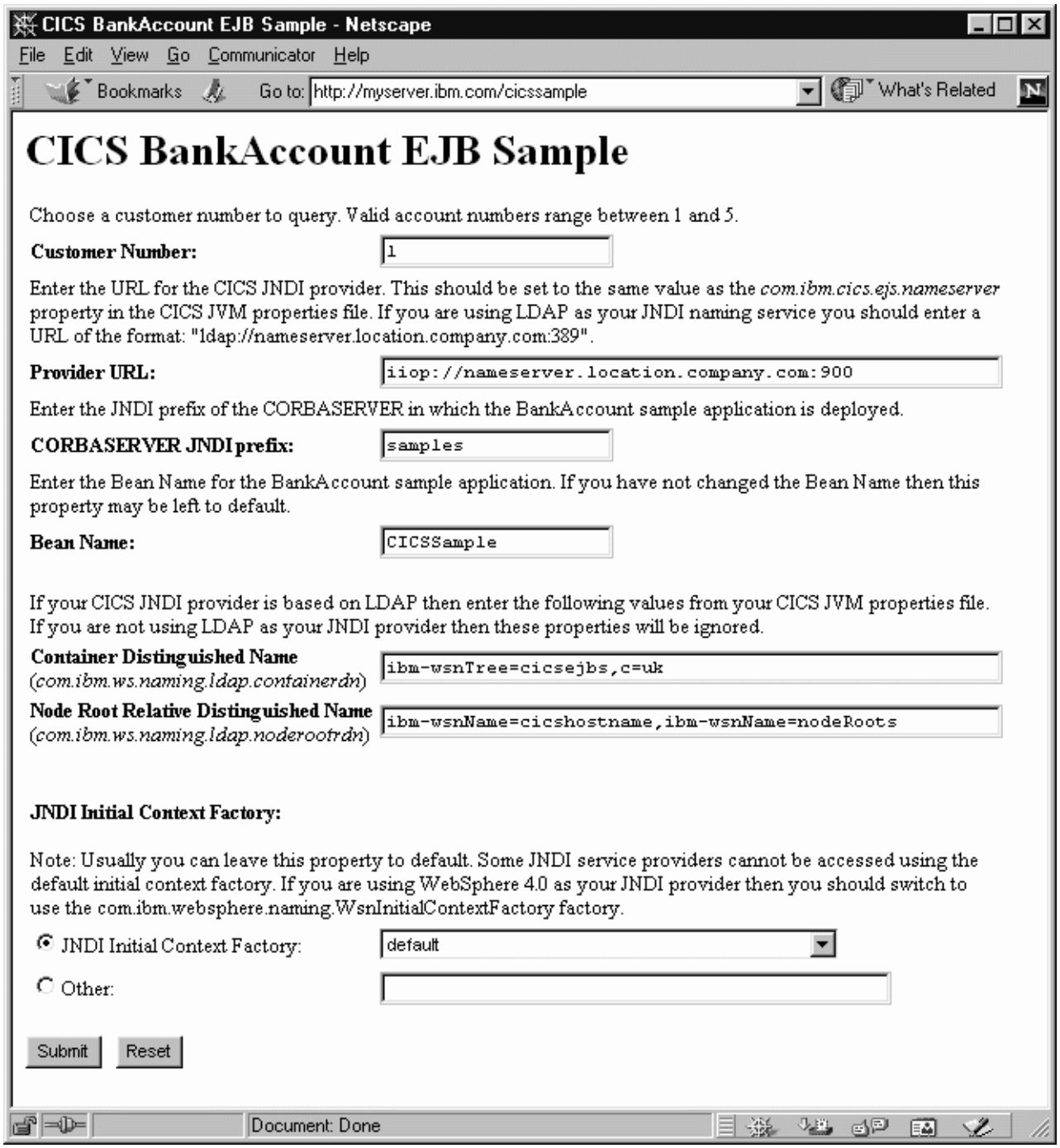


Figura 21. Pantalla de inicio de la aplicación de ejemplo EJB Bank Account

3. Escriba un número de cliente. (Mediante los datos de DB2 provistos, los números de cliente válidos son del 1 al 5).
4. Compruebe que los campos `Provider URL:`, `CORBASERVER JNDI prefix:`, `Bean Name:`, `Container Distinguished Name:`, `Node Root Relative Distinguished Name:` y `JNDI Initial Context Factory` contienen valores que son válidos para la instalación. Si no es así, sobrescríbalos del siguiente modo:

   `Provider URL:`
   Escriba el URL y el número de puerto del servidor de nombres en el que se

publica el enterprise bean. (Estos los especifica la propiedad **-Dcom.ibm.cics.ejs.nameserver** del archivo de propiedades de JVM). Por ejemplo:

- Si utiliza un servidor de COS Naming Server con un URL de `mycosns.ibm.com` y un número de puerto de 900, especifique `"iiop://mycosns.ibm.com:900"`.
- Si utiliza un servidor de nombres LDAP con un URL de `myldapns.ibm.com` y un número de puerto de 389, especifique `"ldap://myldapns.ibm.com:389"`.
- Si utiliza el CORBA Object Services Naming Directory Server incluido con WebSphere Application Server versión 5 o posteriores, con un URL de `mycosns.ibm.com` y un número de puerto de 2809, especifique:

  ```
  -Dcom.ibm.cics.ejs.nameserver=iiop://mycosns.ibm.com:2809/domain/legacyRoot
  ```

Para obtener información detallada sobre cómo especificar la ubicación del servidor de nombres, consulte la descripción de la propiedad **-Dcom.ibm.cics.ejs.nameserver** en "Propiedades del sistema de la JVM" en la página 118.

CORBASERVER JNDI prefix:
Escriba el prefijo JNDI del CorbaServer. Si utiliza la definición de CORBASERVER incluida en DFH$EJB, no tiene que cambiar el valor predeterminado de "`samples`".

Bean name:
Escriba el nombre del enterprise bean utilizado por el ejemplo, como se define en el descriptor de despliegue que hay en el archivo `SampleEJB.jar` incluido. *A no ser que haya renombrado el bean, no es necesario cambiar el valor predeterminado de "CICSSample".*

Container Distinguished Name:
Si utiliza un servidor de nombres LDAP, escriba el nombre distinguido de la raíz del espacio de nombres del sistema LDAP, como se lo proporcione su administrador de LDAP. (El nombre distinguido de la raíz del espacio de nombres del sistema LDAP se especifica mediante la propiedad del sistema **-Dcom.ibm.ws.naming.ldap.containerdn**). *Si utiliza un servidor de nombres COS, el valor de este campo se ignora.*

Node Root Relative Distinguished Name:
Si utiliza un servidor de nombres LDAP, escriba el nombre distinguido de la raíz del nodo LDAP, como se lo proporcione su administrador de LDAP. (El nombre distinguido de la raíz del espacio de nombres del sistema LDAP se especifica mediante la propiedad **-Dcom.ibm.ws.naming.ldap.noderootrdn**). *Si utiliza un servidor de nombres COS, el valor de este campo se ignora.*

JNDI Initial Context Factory:
Seleccione la fábrica de contexto inicial de JNDI adecuada de la lista desplegable. Si el servidor de aplicaciones web es WebSphere, la fábrica que se utilizará depende de:

- La versión de WebSphere que utilice.
- La ubicación WebSphere: es decir, si está en una plataforma distribuida como Windows NT o en una plataforma de host como z/OS
- El tipo de servidor de nombres que utiliza: nombres COS o LDAP

Tabla 19 en la página 304 muestra la fábrica de contexto inicial correcta para especificar, si el servidor de aplicaciones web es WebSphere.

Tabla 19. Definición de la fábrica de contexto inicial con arreglo a la versión y a la ubicación de WebSphere y al tipo del servidor de nombres

Versión de WebSphere	Ubicación del servidor de aplicaciones web.	Tipo de servidor de nombres.	Fábrica de contexto inicial que utilizar.
3.5	Distribuido	COS	`com.ibm.ejs.ns.jndi.CNInitialContextFactory`
3.5	Distribuido	LDAP	`com.ibm.jndi.LDAPCtxFactory`
3.5	z/OS	COS	`com.sun.jndi.cosnaming.CNCtxFactory`
3.5	z/OS	LDAP	`com.sun.jndi.ldap.LdapCtxFactory`
4 o posterior	Distribuido	COS o LDAP	`com.ibm.websphere.naming.WsnInitialContextFactory`
4 o posterior	z/OS	COS	`com.sun.jndi.cosnaming.CNCtxFactory`
4 o posterior	z/OS	LDAP	`com.sun.jndi.ldap.LdapCtxFactory`

Si el servidor de aplicaciones web no es WebSphere, elija el valor adecuado de la lista desplegable:

Nota: La lista desplegable contiene varias clases de fábrica de contexto inicial, más un elemento de lista "default" (predeterminado). La aplicación de ejemplo asigna el valor del elemento de lista predeterminado del siguiente modo:

a. Si se encuentra la clase `com.ibm.websphere.naming.WsnInitialContextFactory` en la vía de acceso de clases Java, el ejemplo la convierte en el elemento predeterminado. Esta clase es un "derivador" que recorta tanto `com.ibm.ejs.ns.jndi.CNInitialContextFactory` como `com.ibm.jndi.LDAPCtxFactory`. La aplicación de ejemplo determina la clase base correcta examinando el tipo de servidor de nombres que ha especificado en el campo **Provider URL**: si el protocolo especificado es "iiop", la aplicación de ejemplo utiliza `com.ibm.ejs.ns.jndi.CNInitialContextFactory`; si es "ldap", utiliza `com.ibm.jndi.LDAPCtxFactory`.

b. Si la clase `com.ibm.websphere.naming.WsnInitialContextFactory` *no* se encuentra en la vía de acceso de clases Java, la aplicación de ejemplo determina la clase base correcta examinando el tipo de servidor de nombres que ha especificado en el campo **Provider URL**: si el protocolo especificado es "iiop", la aplicación de ejemplo utiliza `com.ibm.ejs.ns.jndi.CNInitialContextFactory`; si es "ldap", utiliza `com.ibm.jndi.LDAPCtxFactory`.

Si ninguno de los valores de la lista desplegable es válido para la instalación, seleccione el botón de selección `Other` y escriba el valor correcto en el campo de texto inferior.

5. Pulse el botón SUBMIT. Esto invoca el servlet y ejecuta la aplicación.

   Si la aplicación está configurada correctamente y los valores de entrada son válidos, las JSP de `SampleResults` visualizan los detalles del cliente en el navegador web. La Figura 22 en la página 305 muestra el resultado de una consulta correcta.

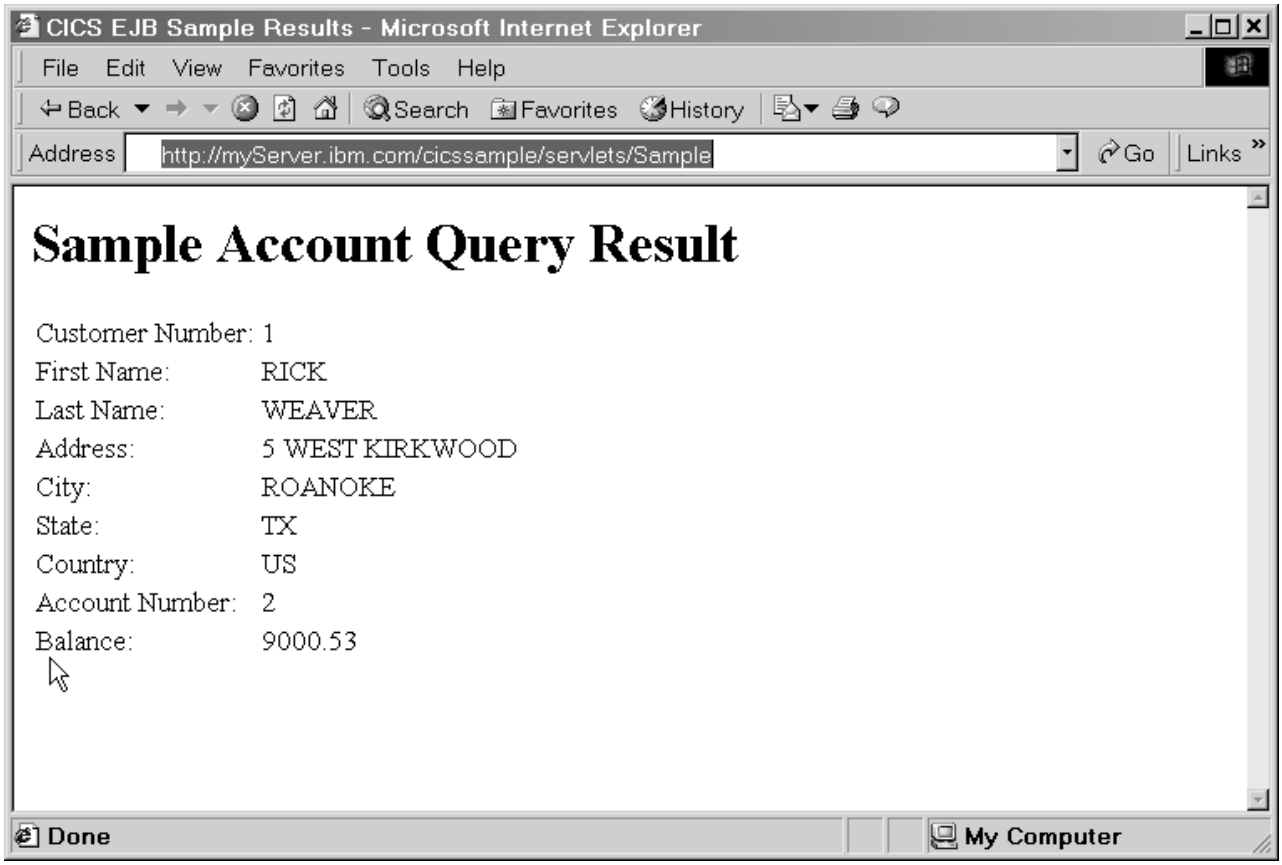


Figura 22. Pantalla de resultados de la aplicación de ejemplo EJB Bank Account

Si la aplicación no está configurada correctamente, o uno o más de los valores de entrada no es válido, las JSP de `SampleError` visualizan un mensaje de error en el navegador web. El archivo `readme.txt` contiene sugerencias y consejos que pueden ayudarle a depurar una aplicación con anomalías.

Una nota sobre las transacciones distribuidas:

Existen diversos protocolos para el soporte de las transacciones distribuidas.

El entorno de CICS enterprise Java soporta solo el protocolo Servicio de transacción de objetos (OTS) de CORBA. Sin embargo, algunos servidores de aplicaciones web compatibles con J2EE (como WebSphere) no utilizan este protocolo o no lo utilizan de forma predeterminada. Es posible configurar WebSphere para que utilice transacciones distribuidas del OTS; para obtener instrucciones detalladas sobre cómo configurar WebSphere para que utilice el OTS, consulte el archivo `readme.txt` incluido con la aplicación de ejemplo Bank Account.

Si hay objetos en su servidor de aplicaciones web que llaman a enterprise beans de CICS dentro del ámbito de contextos de transacción existentes, debe configurar dicho servidor de aplicaciones web para que utilice el OTS de CORBA.

Cambio de la aplicación de ejemplo para utilizar transacciones distribuidas:

Puede intentar este ejercicio para probar si su servidor de aplicaciones web J2EE es completamente compatible con CICS o no.

Acerca de esta tarea

De forma predeterminada, la aplicación de ejemplo EJB Bank Account no está configurada para utilizar transacciones distribuidas. Sin embargo, esto puede cambiarse. El servlet `SampleServlet` contiene código de ejemplo, con comentarios, para activar las transacciones demarcadas por el cliente. (El archivo de origen `SampleServlet.java` se encuentra en el archivo `CicsSample.ear`).

Para activar las transacciones demarcadas por el cliente:
1. Elimine los comentarios del código relacionado con la transacción en `SampleServlet.java`.
2. Vuelva a compilar el servlet `SampleServlet`.
3. Instale la copia actualizada del servlet en el servidor de aplicaciones web.

Si define la aplicación de ejemplo para utilizar transacciones demarcadas por el cliente pero el servidor de aplicaciones web J2EE no soporta transacciones del OTS puras (o no está configurado para utilizarlas), al ejecutar la aplicación de ejemplo, CICS genera una excepción `org.omg.CORBA.INVALID_TRANSACTION`. Esto se debe a que se envió un contexto de transacción pero CICS no pudo utilizarlo.

Cambio del atributo de transacción del enterprise bean:

Tal vez desee modificar el atributo de transacción del enterprise bean (en el descriptor de despliegue) de 'Supports' (soporta) a 'Mandatory' (obligatorio).

Si lo hace, CICS permite invocar el método remoto del bean únicamente si en la llamada se pasa un contexto de transacción del OTS existente desde el entorno del cliente.

Por otro lado, si deja el atributo de transacción del enterprise bean definido como 'Supports', CICS enlaza la invocación del método con el contexto de transacción del cliente, si este contexto existe; en caso contrario, el método se ejecuta en una transacción atómica y no se propaga un nuevo contexto de transacción al llamar a otros beans.

Para cambiar el atributo de transacción, puede utilizar el Assembly Toolkit (ATK), que se describe en la *Guía de operaciones y programas de utilidad de CICS*. Tras haber modificado el atributo de transacción, debe hacer lo siguiente para que el cambio sea efectivo:
1. Almacene el archivo `SampleEJB.jar` actualizado en el directorio de recogida (sobrescribiendo la versión anterior).
2. Emita un mandato `CEMT CORBASERVER(corbaserver_name) PERFORM SCAN`.

Si define el atributo de transacción como 'Mandatory' pero no actualiza el servlet para que utilice transacciones demarcadas por el cliente, al ejecutar la aplicación de ejemplo CICS emite una `javax.transaction.TransactionRequiredException`. Esto se debe a que no se ha enviado ningún contexto de transacción.

Una nota sobre la conversión de datos:

Para representar datos de texto, los programas Java siempre utilizan el juego de caracteres Unicode, mientras que los programas CICS TS utilizan EBCDIC.

Cuando un programa o un enterprise bean Java llama a un programa de servidor de CICS TS, cualquier valor de texto en el área de comunicaciones del programa de servidor debe convertirse de Unicode a EBCDIC en la entrada y de EBCDIC a

Unicode en la salida. El enterprise bean de la aplicación de ejemplo EJB Bank Account utiliza el CCI Connector for CICS TS, que maneja esta conversión de datos automáticamente; consulte “Conversión de datos del CCI Connector for CICS TS” en la página 346.

Nota: Solo se convierten de EBCDIC a Unicode los datos de texto devueltos por el programa COBOL DFH0CSTD. (No es necesaria ninguna conversión para el programa de servidor DFH0ACTD, ni en la entrada para DFH0CSTD, ya que no hay ningún valor de texto en las áreas de comunicaciones).

Escritura de enterprise beans

Puede escribir beans de sesión. Las interfaces utilizadas con estos beans se correlacionan con los servicios y recursos de CICS y los beans se pueden trasladar a cualquier otro servidor compatible con EJB.

Los beans de sesión utilizan las interfaces definidas por la especificación *Enterprise JavaBeans Specification, versión 1.1*. Para descargar la especificación, vaya al sitio web Oracle Technology Network Java y busque *"Enterprise JavaBeans specification"* para localizar la página web de especificaciones.

También puede escribir beans de sesión que utilicen las clases de JCICS para acceder a servicios y recursos de CICS directamente. Estos beans se pueden mover solo a otros servidores EJB de CICS.

CICS no soporta los beans de entidad; es decir, no se pueden ejecutar beans de entidad en un servidor EJB de CICS. (Un bean de sesión o un programa que se ejecute en un servidor EJB de CICS puede comunicarse con un bean de entidad que se ejecute en un servidor EJB que no sea de CICS).

Puede escribir sus propios beans en una estación de trabajo utilizando un entorno de desarrollo integrado (IDE) que soporte la *Enterprise JavaBeans Specification, versión 1.1*.

Al desarrollar nuevos enterprise beans y programas de Java para CICS, debe utilizar un entorno de desarrollo de aplicaciones que soporte Java 2 en el nivel de SDK 5.0. **No** debe:

- Utilice cualquier llamada a API que esté soportada solo por una una versión más nueva de el Java SDK que el que soporta CICS.
- Utilice las características admitidas solo por una versión posterior de la *Enterprise JavaBeans Specification* que la admitida por CICS. (Actualmente, CICS soporta la *Enterprise JavaBeans Specification, versión 1.1*.)

Todos los enterprise beans desarrollados según la especificación EJB 1.0 deben migrarse al nivel de especificación EJB 1.1 utilizando las herramientas de desarrollo incluidas; consulte “Las herramientas de despliegue para enterprise beans en un sistema CICS” en la página 321.

“Codificación de un bean de sesión” en la página 308 brinda un ejemplo de los pasos necesarios para escribir un bean de sesión sin utilizar un IDE.

Puede utilizar el CCI Connector for CICS TS para compilar enterprise beans que aprovechen los programas CICS ya existentes. Consulte “El CCI Connector for CICS TS” en la página 337 para obtener una descripción del CCI Connector for CICS TS y cómo utilizarlo.

Preparación de beans para su ejecución

El proceso de instalación y preparación de un enterprise bean para su ejecución se conoce como **despliegue**.

CICS proporciona herramientas basadas en estación de trabajo para gestionar el despliegue de enterprise beans en el entorno CICS del host.

La estación de trabajo y los componentes de WebSphere de las herramientas de despliegue se proporcionan como un conjunto de paquetes de asistente de instalación. Puede descargar estos paquetes desde el sistema z/OS o ejecutarlos desde el CD que se le haya proporcionado en la estación de trabajo de destino.

Consulte “Despliegue de enterprise beans” en la página 321 para obtener una descripción del proceso de despliegue y “Utilización de herramientas de despliegue de CICS para enterprise beans” en la página 322 para obtener orientación sobre cómo utilizar las herramientas.

Codificación de un bean de sesión

Esta sección describe cómo codificar un bean de sesión muy simple.

Cuando haya completado los pasos de esta sección, tendrá un archivo JAR listo para su despliegue. Consulte “Despliegue de enterprise beans” en la página 321 para obtener una descripción del proceso de despliegue y de las herramientas disponibles para ayudarle.

El bean de ejemplo que se muestra aquí simula una ruleta en un casino. La ruleta es un bean de sesión con estado que contiene dos campos con estado. El primer campo es el número actual en el que está la ruleta; el segundo campo es la cantidad de crédito que aún le queda al jugador para apostar. El cliente crea una ruleta, y, como opción, especifica la cantidad de dinero que apostar (de forma predeterminada, 100 dólares si no se proporciona la cantidad). El cliente puede realizar apuestas sobre el color que saldrá y la ruleta gira e indica al apostante si ha ganado o no. Luego el cliente puede recoger las ganancias o seguir apostando.

Hay tres elementos que debe codificar:

1. “Codificación de la interfaz inicial”.
2. “Codificación de la interfaz remota” en la página 309.
3. “Codificación de la implementación del bean” en la página 309.

Luego tiene que compilar y empaquetar el programa:

1. “Compilación del código” en la página 311
2. “Empaquetado del código” en la página 311

Codificación de la interfaz inicial:

La interfaz inicial para un bean amplía la interfaz javax.ejb.EJBHome. Define uno o más métodos create que el programa cliente puede llamar para crear una instancia de bean.

En el caso de beans de sesión sin estado, tiene que haber exactamente un método create sin ningún parámetro. Los beans de sesión con estado pueden sobrecargar el método create con distintas variantes que asumen combinaciones distintas de parámetros. El bean RouletteWheel es un bean de sesión con estado. Sobrecargamos create para que podamos especificar la cantidad de crédito que tenemos en una instancia de ruleta cuando se crea:

```
package casino;

     public interface RouletteWheelHome extends javax.ejb.EJBHome {

       public RouletteWheel create()
         throws javax.ejb.CreateException, javax.ejb.EJBException;

       public RouletteWheel create(int dollars)
         throws javax.ejb.CreateException, javax.ejb.EJBException;
     }
```

Codificación de la interfaz remota:

La interfaz remota para un bean amplía la interfaz javax.ejb.SessionBean. La interfaz remota define los métodos de negocio reales que un programa cliente puede llamar en una instancia de bean individual.

```
package casino;

     public interface RouletteWheel extends javax.ejb.EJBObject {

       // Realice una apuesta a "rojo" o "negro" con la cantidad especificada,
       // el valor de retorno indica al interlocutor si la apuesta tuvo
       // éxito o no.
       public String bet(String bet,int amount) throws javax.ejb.EJBException;

       // Compruebe el estado actual de la ruleta.
       public String getCurrentStatus() throws javax.ejb.EJBException;

       // Cobre las ganancias de la ruleta (si las hay).
       public int collectWinnings() throws javax.ejb.EJBException;

     }
```

Codificación de la implementación del bean:

Esta clase implementa los métodos de negocio definidos en la interfaz remota del bean.

También define algunos métodos estándar que se declaran abstractos en SessionBean y, por lo tanto, estos métodos se deben implementar para completar la implementación del bean. Por último, como sobrecargamos el método create en la interfaz inicial, debemos proporcionar métodos ejbCreate coincidentes en la implementación del bean que acepten los mismos conjuntos de parámetros. Esto se debe a que la clase de implementación del bean es el único lugar en el que se coloca el código de bean. La implementación de la interfaz inicial que definimos en “Codificación de la interfaz inicial” en la página 308 se genera mediante las herramientas, por lo que si necesitamos implementar un método create sobrecargado, debemos hacerlo aquí:

```
package casino;

     import java.util.Random;
     import javax.ejb.*;

     public class RouletteWheelBean implements SessionBean {

       // Código necesario para completar la definición de interfaz de SessionBean.

       private SessionContext ctx = null;

       public void ejbActivate() throws javax.ejb.EJBException {}
       public void ejbPassivate() throws javax.ejb.EJBException {}
       public void ejbRemove() throws javax.ejb.EJBException {}
```

```
public SessionContext getSessionContext() { return ctx; }
public void setSessionContext(SessionContext ctx) throws
  javax.ejb.EJBException { this.ctx = ctx;
}

////////////////////////////
// La información de estado del bean.
       private int wheelValue;

private int currentCredit;

//////////////////////
// Nuestros métodos create.

public void ejbCreate() throws javax.ejb.EJBException, CreateException {
  currentCredit = 100;
  wheelValue = ((int)System.currentTimeMillis())%37;
}

public void ejbCreate(int credit) throws javax.ejb.EJBException,
  CreateException { currentCredit = credit;
  wheelValue = ((int)System.currentTimeMillis())%37;
}

////////////////////////////////////////////////////////////////////////
// Implementaciones de los métodos remotos que el cliente puede llamar en una instancia.

//
// Realice una apuesta, a "rojo" o "negro", por la cantidad especificada.
// Luego simule el giro de la ruleta y construya una serie de respuesta
// que indique el resultado al interlocutor.
//
public String bet(String color,int amount) throws javax.ejb.EJBException {

  if (!color.equalsIgnoreCase("rojo") && !color.equalsIgnoreCase("negro"))
    return new String("Solo puede apostar a rojo o negro.");

  if (amount > currentCredit)
    return new String("¡Solo tiene "+currentCredit+"$!");

  // Utilice el valor de ruleta actual como semilla del número aleatorio.
  Random randomizer = new Random((long)wheelValue);

  // Haga girar la ruleta.
  wheelValue = Math.abs(randomizer.nextInt()) % 37;

  // Construya una respuesta.
  StringBuffer result =
    new StringBuffer("Número: "+wheelValue+" Color: "+color(wheelValue)+"\n");

  // ¿El interlocutor ganó?
  if (color(wheelValue).equalsIgnoreCase(color)) {
    currentCredit+=(amount*2);
    result.append("¡Muy bien! Ganó $");
    result.append((amount*2));
  } else {
    currentCredit -= amount;
    result.append("¡Mala suerte! Perdió $");
    result.append(amount);
  }
  result.append(", ahora tiene $");
  result.append(currentCredit);
  return result.toString();
}


//
```

```
  // Devuelva el estado actual de esta instancia de ruleta.
  // El número y el color
  // en los que está actualmente y la cantidad de crédito que aún tiene el cliente para apostar.
  //
  public String getCurrentStatus() throws javax.ejb.EJBException {
    return new String("Número:"+wheelValue+" Color:"+color(wheelValue)+"
    Tiene $"+currentCredit);
  }


  //
  // Permita al cliente cobrar sus ganancias y luego ponga a cero el crédito, de forma
  // que no se puedan cobrar dos veces.
  //
  public int collectWinnings()throws javax.ejb.EJBException {
    int winnings = currentCredit;
    currentCredit = 0;
    return winnings;
  }


  //
  // Convierta un número de la ruleta en un color.
  //
  private String color(int value) {
    if (value == 0) return "Verde";
    if (value % 2 == 0) return "Negro";
    return "Rojo";
  }

}
```

Compilación del código:

Todo lo que necesita además del SDK base es el archivo JAR que contiene las interfaces `javax.ejb`.

Esto está disponible como `ejb11.jar` en el directorio `standard/ejb/1_1` de la instalación Java. Si añade `ejb11.jar` a su variable CLASSPATH, debe ser capaz de compilar las clases y las interfaces descritas.

Empaquetado del código:

Las clases compiladas deben empaquetarse en un archivo JAR listo para su despliegue.

Si asumimos que los archivos de clase están en el subdirectorio casino, se puede utilizar el siguiente mandato jar:

```
jar -cvf casino.jar casino\*.class
```

Escritura del programa cliente

Un programa cliente es cualquier programa que llama a un enterprise bean.

Puede ser:

1. Otro enterprise bean, un JavaBean, un programa Java o un objeto que se ejecute en el mismo CICS
2. Un enterprise bean, un JavaBean, un programa Java o un objeto que se ejecute en otro CICS
3. Un enterprise bean, un JavaBean, un programa Java o un objeto que se ejecute en un sistema o una estación de trabajo que no sea CICS

El cliente obtiene referencias a los inicios de bean de los enterprise beans que quiere llamar utilizando el espacio de nombres JNDI que comparte con el entorno de servidor de CICS.

Creación de referencias a objeto en el espacio de nombres:

Para crear referencias a objeto, tiene que publicar los beans que estén instalados en su región CICS.

Acerca de esta tarea

Hay dos formas de hacer esto:

1. Emita PERFORM DJAR(XXXX) PUBLISH en el sistema CICS del servidor. Para ello, puede utilizar cualquiera de los siguientes métodos:
   - CEMT
   - CICSPlex SM
   - Una aplicación CICS

   Para cada bean instalado desde el DJAR nombrado, se publica una referencia a objeto en el servidor de directorios de denominación. Consulte “Definición de servidores de nombres” en la página 395 para obtener información sobre el uso de servidor de nombres.
2. Si ha instalado varios DJAR en un único CORBASERVER, puede utilizar el mandato PERFORM CORBASERVER(XXXX) PUBLISH para publicar cada bean instalado actualmente en ese CORBASERVER. El subcontexto del espacio de nombres en el que aparecerán las referencias a objeto los determina el prefijo JNDI definido en la definición de recurso CORBASERVER en el que se instaló el DJAR.

La retracción nunca se realiza de forma implícita. La forma recomendada de 'anular la publicación' de beans es emitir PERFORM DJAR(XXXX)/ CORBASERVER(XXXX) RETRACT. Si se descarta un DJAR o un CORBASERVER, las referencias a objeto de bean seguirán existiendo en el espacio de nombres, aunque serán inutilizables para el cliente, ya que los beans reales ya no existen en CICS. Es posible reinstalar un DJAR y retirar dichas referencias.

Utilización de JNDI para obtener referencias de bean:

La Java Naming and Directory Interface (JNDI) define una interfaz de programación de aplicaciones (API) especificada en el lenguaje de programación Java que proporciona la función de denominación y directorio para programas Java.

También define una interfaz de proveedor de servicios (SPI) que permite conectar varios controladores de servicio de directorio y de denominación. La Figura 23 en la página 313 ilustra esto mostrando un gestor de denominación que interacciona con una aplicación Java mediante la API de JNDI y con varios servidores de nombres mediante la SPI de JNDI.

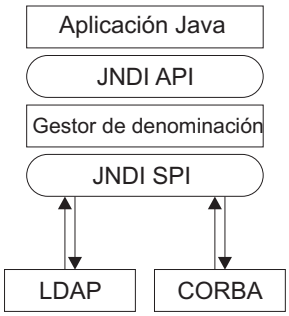


Figura 23. Estructura de JNDI

Las interfaces de JNDI se describen en el siguiente sitio web: http://www.oracle.com/technetwork/java/index.html.

Después de que el administrador del sistema del servidor haya registrado un enterprise bean en un servidor de nombres, una aplicación cliente puede utilizar la interfaz de JNDI para localizar la interfaz inicial del mismo.

Configure un servidor de nombres adecuado que soporte la Java Naming and Directory Interface (JNDI) versión 1.2 y defina su ubicación para CICS. Para obtener más información, consulte "Configuración de un servidor LDAP" en la página 397 y "Configuración de un CORBA Object Services Naming Directory Server" en la página 407. Para obtener detalles de las propiedades de JVM que son necesarias, consulte "Propiedades del sistema de la JVM" en la página 118.

Escritura de un programa cliente para que utilice LDAP:

CICS Transaction Server soporta LDAP. Pueden ser necesarios algunos cambios en los programas cliente para permitir que un programa cliente encuentre los inicios de los bean publicados desde una región CICS.

Un cliente LDAP debe utilizar la fábrica de contexto WebSphere o la fábrica de contexto LDAP suministrada por Java. La ventaja de utilizar la fábrica de contexto WebSphere es que esta comprende automáticamente el espacio de nombres del sistema (es decir, el espacio de nombres estructurado en el servidor LDAP en el que CICS publica sus inicios de bean). Sin embargo, esta fábrica de contexto tiene distintas dependencias, por lo que no es el cliente más ligero. La fábrica de contexto suministrada por Java no tiene otras dependencias que el IBM Developer Kit for the Java Platform base y, por lo tanto, es muy ligero. Sin embargo, no comprende el espacio de nombres del sistema, por lo que es necesario negociarlo mediante programación, pero CICS proporciona algunos métodos de programa de utilidad para ayudarle a hacerlo.

Estas alternativas se demuestran mejor mediante ejemplos.

Fábrica de contexto WebSphere:

Este es un ejemplo de código fuente de cliente que utiliza la fábrica de contexto de WebSphere para localizar el inicio de un bean HelloWorld.

```
import org.omg.CORBA.ORB;
import java.io.*;
import javax.naming.*;
import examples.helloworld.*;
import java.util.*;
```

```
    public class WASNamingClient {
       public static void main(String[] argv) {
          try {

    // Defina las propiedades necesarias.
          Properties prop = new Properties();

    // Estos cuatro son valores *fijos*, nunca tiene que cambiarlos.

      prop.put(Context.INITIAL_CONTEXT_FACTORY,
           "com.ibm.websphere.naming.WsnInitialContextFactory");

      prop.put("com.ibm.websphere.naming.namespaceroot","bootstraphostroot");
      prop.put("com.ibm.ws.naming.ldap.config","local");
      prop.put("com.ibm.ws.naming.implementation","WsnLdap");

    // Estos dos dependen de los valores de su servidor y deben coincidir con los valores de la región CICS.

      prop.put("com.ibm.ws.naming.ldap.containerdn","ibm-wsnTree=WASNaming,c=us");
      prop.put("com.ibm.ws.naming.ldap.noderootrdn",
  "ibm-wsnName=legacyroot,ibm-wsnName=PLEX2,ibm-wsnName=domainRoots");

  // Finalmente, en lugar de com.ibm.cics.ejs.nameserver,
  // defina com.ibm.ws.naming.ldap.masterurl como el servidor LDAP de destino.

        prop.put("com.ibm.ws.naming.ldap.masterurl","ldap://wibble.example.com:389");

        InitialContext ctx = new InitialContext(prop);
        org.omg.CORBA.Object obj =
             (org.omg.CORBA.Object)ctx.lookup("samples/HelloWorld");

        HelloWorldHome hhome =
             (HelloWorldHome)javax.rmi.PortableRemoteObject.narrow
             (obj,HelloWorldHome.class);

        System.out.println("iHelloWorldHome encontrado correctamente!");
        HelloWorld hello = hhome.create();
        System.out.println(hello.sayHello());

   } catch (Exception e) {
     System.err.println("Excepción al buscar y llamar al bean HelloWorld:");
     e.printStackTrace();
    }
  }
}
```

Como se indica en los comentarios, las primeras cuatro propiedades son fijas, en las tres restantes coinciden los valores para la región CICS (aunque la propiedad **-Dcom.ibm.cics.ejs.nameserver** se ha convertido en com.ibm.ws.naming.ldap.masterurl). Sin embargo, la fábrica de contexto de WebSphere tiene dependencias con componentes de WebSphere, por lo que para ejecutarla desde la línea de mandatos debe ejecutar un script para configurar el entorno adecuadamente.

El script `DFHWAS4Setup.bat` es un script de la línea de mandatos que se incluye con CICS. Se puede descargar desde el subdirectorio utils en el área de z/OS UNIX en la que CICS está instalado. Debe ejecutarse en un sistema que tenga instalado WebSphere, porque depende de que la variable de entorno WAS_HOME se defina para apuntar a la ubicación en la que se ha instalado WebSphere; por ejemplo, `c:\WebSphere\AppServer`. Cuando se haya ejecutado el script, debe ampliar su variable CLASSPATH aún más para incluir el código del lado del cliente necesario para el enterprise bean. Para el ejemplo anterior, este es `HelloWorld.jar`; luego el

código anterior se puede compilar y ejecutar. (El código de ejemplo asume que el inicio se publica en un CorbaServer cuyo prefijo de JNDI sea *samples*).

En CICS definimos **-Dcom.ibm.cics.ejs.nameserver =** <*nombre_host*>, pero en este programa cliente definimos **com.ibm.ws.naming.ldap.masterurl =** <*nombre_host*>. CICS comprende la primera forma, WebSphere comprende la segunda.

Fábrica de contexto LDAP suministrada con Java:

Desde el punto de vista de la configuración de IBM Developer Kit for the Java Platform, es mucho más fácil utilizar la fábrica de contexto LDAP suministrada con Java, ya que se incluye en el IBM Developer Kit for the Java Platform base y no tiene ninguna dependencia fuera del mismo.

Sin embargo, como esta fábrica de contexto no comprende la estructura del espacio de nombres que existe en cualquier servidor LDAP configurado para WebSphere, puede ser más difícil para el programador de aplicaciones cliente. CICS brinda algunas funciones de ayudante para el espacio de nombres que reducen esta complejidad extra. La clase com.ibm.cics.portable.CICSNameSpaceHelper se incluye en `CICSEJBClient.jar`. Este archivo JAR está disponible en el subdirectorio `utils` del área de z/OS UNIX en la que CICS está instalado.

Aquí tiene un ejemplo de la utilización de esta clase:

```
import org.omg.CORBA.ORB;
import java.io.*;
import examples.helloworld.*;
import javax.naming.*;
import javax.naming.directory.*;
import java.util.*;
import com.ibm.cics.portable.CICSNameSpaceHelper;

public class SUNNamingClient {

public static void main(String[] argv) {

   try {
      Hashtable env = new Hashtable();

// Configure las dos primeras propiedades obvias, la fábrica LDAP y el servidor LDAP suministrados
   con Java.
      env.put(Context.INITIAL_CONTEXT_FACTORY, "com.sun.jndi.ldap.LdapCtxFactory");
      env.put(Context.PROVIDER_URL,   "ldap://wibble.example.com:389");
// Estos dos valores coinciden con los valores procedentes del sistema CICS.
      env.put("com.ibm.ws.naming.ldap.containerdn",  "ibm-wsnTree=WASNaming,c=us");
      env.put("com.ibm.ws.naming.ldap.noderootrdn",
            "ibm-wsnName=legacyroot,ibm-wsnName=PLEX2,ibm-wsnName=domainRoots");

// Utilice el método de ayudante LDAPSNSLookup para negociar el espacio de nombres del sistema WebSphere
// en wibble.example.com y localice nuestro bean HelloWorld. "samples"
// es el prefijo de JNDI en el CorbaServer de CICS que publicó el bean HelloWorld.
      org.omg.CORBA.Object obj =
      CICSNameSpaceHelper.LDAPSNSLookup(env,"samples/HelloWorld");

    HelloWorldHome hhome =
            (HelloWorldHome)javax.rmi.PortableRemoteObject.narrow
            (obj,HelloWorldHome.class);

    System.out.println("¡Inicio de HelloWorld encontrado correctamente!");
    Hello hello = hhome.create();
      System.out.println(hello.sayHello());
 } catch (Exception e) {
   System.err.println("Excepción al buscar y llamar al bean HelloWorld:");
```

```
        e.printStackTrace();
    }
  }
}
```

Está utilizando el código de LDAP suministrado con Java, que comprende la propiedad `providerURL`, en lugar de la propiedad `masterurl` utilizada en el ejemplo de la fábrica de contexto de WebSphere.

La clase de ayudante `CICSNameSpaceHelper` también puede funcionar con otras fábricas de contexto. Tenga en cuenta que la sintaxis del nombre que se pasa a `LDAPSNSLookup` es sintaxis de JNDI `a/b/c/d`.

Escritura de un programa cliente para que utilice denominación COS: El siguiente ejemplo muestra un programa cliente, `Gambler.java`, que trabaja con el bean `RouletteWheel` desarrollado en "Codificación de un bean de sesión" en la página 308. Cuando se obtiene una referencia de bean de un espacio de nombres de denominación COS, se deben realizar distintas operaciones antes de que el cliente pueda utilizar esa referencia. Estas operaciones son las mismas para la mayoría de los programas cliente, por lo que se recopilan en la clase de programa de utilidad `EJBUtils`. Esta clase de programa de utilidad la utiliza el programa cliente `Gambler`.

EJBUtils.java:

Esta es la implementación de la clase de programa de utilidad `EJBUtils`.

```
import javax.naming.*;
import java.util.Hashtable;

class EJBUtils {

  public static Object jndi_lookup(String name, Class resultClass) {

    // Configure el entorno para crear el contexto inicial.
    Hashtable env = new Hashtable(11);

    // Defina el servidor de nombres; consulte la nota 1 más adelante.
    env.put(Context.PROVIDER_URL,
      "iiop://wibble.example.com:900");

    // Defina la fábrica de contexto inicial; consulte la nota 2.
    env.put(Context.INITIAL_CONTEXT_FACTORY,
      "com.sun.jndi.cosnaming.CNCtxFactory");

    try {

      // Cree el contexto inicial.
      Context ctx = new InitialContext(env);

      // Busque el objeto.
      Object tempObject = ctx.lookup(name);

      // Reduzca la búsqueda a la clase solicitada.
      return javax.rmi.PortableRemoteObject.narrow(tempObject,resultClass);

    } catch (NamingException ne) {
      System.err.println("EJBUtils.jndi_lookup() failed:");
      ne.printStackTrace();
    }
    return null;
```

```
    }
  }
```

Nota:

1. Aquí definimos el servidor de nombres que se utilizará para la búsqueda de beans como "iiop://wibble.example.com:900". Este valor debe ser el nombre de su servidor de nombres y coincidir con el **-Djava.naming.provider.url** que se definiera en el archivo de propiedades de JVM de CICS, de forma que el cliente busque el bean en el servidor de nombres en el que fue publicado por CICS. Consulte “Definición de servidores de nombres” en la página 395 para obtener información sobre el uso de servidores de nombres.
2. Aquí definimos la fábrica de contexto inicial para el entorno de cliente. Debe definirla al valor necesario para el entorno de cliente. El ejemplo muestra el valor que definiría al utilizar el ORB incluido con el IBM SDK. Si su cliente es una aplicación Java o un enterprise bean que se ejecuta en CICS Transaction Server para z/OS, versión 2, no debe especificar aquí una fábrica de contexto inicial, sino que debe permitir que tenga el valor predeterminado de `com.ibm.websphere.naming.wsnInitialContextFactory`.

Gambler.java:

Esta es la implementación del programa cliente de ejemplo `Gambler.java`.

```
import org.omg.CORBA.ORB;
import java.io.*;
import casino.*;

public class Gambler {

  public static void main(String[] argv) {

    try {

      System.out.println("Jugador\n");

      System.out.println("Buscando inicio de RouletteWheel");
      RouletteWheelHome wheelHome =
        (RouletteWheelHome)
        EJBUtils.jndi_lookup("cics/ejbs/RouletteWheel",
                             RouletteWheelHome.class);
        //
        // Consulte la nota 1.
        //
      System.out.println("Creando una nueva ruleta.");
      RouletteWheel wheel = wheelHome.create();

      System.out.println("");
      System.out.println("¡Apostando $50 al rojo!");
      System.out.println(wheel.bet("rojo",50));

      System.out.println("");
      System.out.println("¡Apostando $20 al negro!");
      System.out.println(wheel.bet("negro",20));

      System.out.println("");
      System.out.println("¡Apostando $20 al rojo!");
      System.out.println(wheel.bet("rojo",20));

      System.out.println("");
      System.out.print("Recogiendo ganancias:$");
      System.out.println(wheel.collectWinnings());
```

```
        System.out.println("");
        System.out.print("Eliminando la ruleta");
        wheel.remove();

      } catch (Exception e) {
        System.err.println("Error al jugar:");
        e.printStackTrace();
      }

    }

  }
```

Nota:

1. El programa cliente Gambler.java busca la ruleta (RouletteWheel) en "cics/ejbs" en el espacio de nombres. Esto significa que el CorbaServer de CICS en el que ha instalado el bean RouletteWheel debe tener un prefijo JNDI de cics/ejbs. Una vez instalado y publicado, RouletteWheel será accesible para el programa cliente.
2. Hay una llamada a `remove` (eliminar) al final de este programa cliente. El bean de la ruleta es con estado y CICS gestiona el estado de cada instancia. A no ser que se llame a `remove`, al finalizar la operación con esa instancia de bean, CICS seguirá almacenándola. El tiempo de espera del bean se puede controlar mediante el parámetro SESSBEANTIME de la definición de recurso CORBASERVER. Esto indica a CICS cuánto debe gestionar el estado de la instancia si no llega ninguna solicitud para utilizar dicha instancia, implementando una especie de recogida de basura. Sin embargo, una práctica de programación recomendada es llamar a `remove` cuando se haya finalizado el trabajo con una instancia, de manera que no se dependa de este tipo de recogida de basura.

Utilización del programa cliente:

Al compilar el programa cliente, la vía de acceso de clases debe definirse con cuidado para incluir el archivo JAR desplegado que procesara antes correctamente con la CICS Jar Development Tool, así como las interfaces javax.ejb para soporte de EJB 1.1, que están disponibles en `ejb11.jar`, en el directorio `standard/ejb/1_1` de la instalación Java.

Una vez compilado, ejecute el cliente con:

```
java Gambler
```

Interoperatividad de transacciones con servidores de aplicaciones web:

Existen varios protocolos para el soporte de transacciones distribuidas. El entorno de CICS enterprise Java soporta solo el protocolo Servicio de transacción de objetos (OTS) de CORBA estándar. Sin embargo, algunos servidores de aplicaciones web compatibles con J2EE (como WebSphere versión 4) no utilizan este protocolo o no lo utilizan de forma predeterminada.

Si hay objetos en su servidor de aplicaciones web que llaman a enterprise beans de CICS dentro del ámbito de contextos de transacción existentes, debe configurar dicho servidor de aplicaciones web para que utilice el OTS de CORBA. Si esto no es posible, su servidor de aplicaciones web no es totalmente compatible con el soporte de CICS enterprise Java. (Si desea conocer un modo de utilizar la aplicación de ejemplo EJB Bank

Account para probar si su servidor de aplicaciones web es totalmente compatible con el soporte de CICS enterprise Java, consulte “Una nota sobre las transacciones distribuidas” en la página 305).

Si su servidor de aplicaciones web es WebSphere Application Server versión 4, tenga en cuenta que, de forma predeterminada, no utiliza el OTS de CORBA estándar, pero puede hacerlo. Si tiene objetos WebSphere que llaman a enterprise beans de CICS dentro del ámbito de contextos de transacción existentes, debe configurar WebSphere para utilizar el OTS de CORBA. (Las versiones de WebSphere Application Server desde la versión 5 en adelante no se ven afectadas por este problema).

Para forzar a que WebSphere Application Server utilice el OTS de CORBA:
1. En la WebSphere Administration Console, seleccione el separador de valores de JVM.
2. Especifique lo siguiente en la sección "System Properties" (Propiedades del sistema):

   ```
   com.ibm.ejs.jts.ControlSet.interoperabilityOnly=true
   com.ibm.ejs.jts.ControlSet.nativeOnly=false
   ```

   Guarde los cambios.
3. Reinicie el servidor de aplicaciones.

Trabajo con las interfaces Handle, HomeHandle y EJBMetaData de EJB

La especificación Enterprise JavaBeans describe cómo un bean de sesión soporta no solo los métodos definidos en su interfaz remota sino también algunos métodos más.

- En la interfaz inicial de EJB hay métodos definidos que pueden ser llamados por un cliente que desee:
  - Bien obtener una referencia “almacenable” a la interfaz inicial (un descriptor de contexto de inicio).
  - O bin obtener los EJBMetaData para el tipo de bean.

  .
- En la interfaz EJBObject hay métodos definidos que pueden ser llamados por un cliente que desee:
  - Bien pbtener la interfaz inicial para el EJB.
  - O bien obtener una referencia “almacenable” al propio objeto (un descriptor de contexto).

La finalidad de los descriptores de contexto es que son serializables: cuando se obtiene un descriptor de contexto para una instancia de bean, este se puede serializar, quizá en un archivo sin formato. Si en algún momento posterior un programa quiere realizar llamadas a la misma instancia, puede deserializar el descriptor de contexto y empezar a llamar a métodos de nuevo. Las implementaciones de los descriptores de contexto y las clases de metadatos son específicas de producto.

En CICS, las implementaciones de las tres interfaces, HomeHandle, Handle y EJBMetaData, son:

- com.ibm.cics.portable.CICSSessionHomeHandle
- com.ibm.cics.portable.CICSSessionHandle
- com.ibm.cics.portable.CICSEJBMetaData

Estas implementaciones se incluyen en el archivo JAR `CICSEJBClient.jar`, que se puede descargar del subdirectorio `utils` en el área de z/OSUNIX en la que CICS está instalado. Este archivo JAR debe incluirse en la variable CLASSPATH de cualquier programa cliente que llame a los métodos especiales descritos anteriormente, para asegurar que este comprende los tipos de objetos devueltos desde el servidor. Por ejemplo, si su variable CLASSPATH no incluye `CICSEJBClient.jar`, un programa cliente que llame a la función **getEJBMetaData** de un enterprise bean puede devolver alguno de los siguientes:

1. Una excepción.
2. Un valor nulo.

El valor preciso devuelto depende de la implementación del intermediario para solicitudes de objetos (ORB) del cliente.

Utilización de EDF con enterprise beans

Para utilizar EDF para probar enterprise beans, debe realizar las siguientes tareas.

Acerca de esta tarea

- Defina el parámetro CEDF como YES en la definición de recurso PROGRAM para DFJIIRP, que se proporciona en el grupo DFHIIOP.
- Defina MAXACTIVE a uno en TRANCLASS(DFHEDFTC).
- Active EDF especificando CEDX (*id_transacción*) en el terminal donde se interrumpirá la transacción. El ID de transacción es el ID de transacción predeterminado CIRP o la transacción especificada en la definición RequestModel.
- Lance el bean.

Comunicación de bean a bean:

Si su bean utiliza comunicación de bean a bean con el mismo identificador de transacción en la misma AOR, definir MAXACTIVE en uno resultará en que la comunicación no se produzca.

Esto se debe a que la ejecución de la segunda transacción se suspenderá en espera de una ranura en la que ejecutarse, y el bean original sufrirá una condición de "timeout" (tiempo de espera excedido). El modo de evitarlo es realizar una de las siguientes acciones:

- Utilice REQUESTMODEL para especificar un ID de transacción exclusivo para cada bean.
- Permita que todos los métodos create utilicen CIRP (el ID de transacción predeterminado) y use REQUESTMODEL para definir un ID de transacción exclusivo para cada conjunto de métodos de negocio.

Nota: Si un bean se ejecuta en un procesador de solicitudes, CICS solo utilizará requestmodels (y, por lo tanto, lanzará una nueva transacción de CICS con el nuevo ID de transacción) si una llamada a método remoto realizada por ese bean no se puede satisfacer en el procesador de solicitudes actual. Una llamada a método no se puede satisfacer de forma local en el procesador de solicitudes actual si:

- Los atributos de transacción del método al que se llama necesitan un contexto de transacción distinto.
- El bean al que se llama está en un CorbaServer distinto.

Despliegue de enterprise beans

Esta sección explica el proceso de despliegue de enterprise beans en un servidor EJB de CICS de forma más detallada.

El concepto del despliegue se presenta en “Visión general del despliegue de enterprise beans” en la página 246.

El término “despliegue” que se utiliza en la especificación de EJB describe una serie de tareas que hacen que los enterprise beans de uno o más archivos JAR estén disponibles para su utilización en un entorno operativo específico (en este caso, un servidor EJB de CICS).

Las herramientas de despliegue para enterprise beans en un sistema CICS

CICS proporciona tres herramientas para ayudarle en el despliegue de enterprise beans en un servidor EJB de CICS.

El Assembly Toolkit (ATK)

El Assembly Toolkit (ATK) es una herramienta general que utilizan varios servidores EJB de IBM, incluido CICS, para crear archivos JAR listos para que el entorno de ejecución los utilice. El Assembly Toolkit for Windows se incluye con WebSphere Application Server.

Para obtener información detallada sobre la utilización del ATK, consulte , en la *Guía de operaciones y programas de utilidad de CICS*.

El gestor de recursos para enterprise beans

El gestor de recursos para enterprise beans es una herramienta web que puede utilizar para realizar ciertas operaciones en los recursos (CORBASERVER y DJAR) que están instalados en CICS para soportar la utilización de enterprise beans. También puede utilizar la herramienta para el diagnóstico de problemas relacionados con EJB, ya que ofrece la capacidad de visualizar cualquier error asociado con definiciones DJAR e indica si los beans de un archivo JAR desplegado se han publicado en el servicio de denominación.

Si desea una descripción completa del gestor de recursos para enterprise beans, consulte la *Guía de operaciones y programas de utilidad de CICS*.

Transacción CREA

CREA es una transacción que proporciona CICS y que se puede utilizar para crear definiciones de recurso REQUESTMODEL para los beans de un archivo JAR desplegado que haya instalado. CREA puede instalar definiciones en un sistema CICS en ejecución mediante mandatos **EXEC CICS CREATE**, o puede grabar las definiciones en el CSD. CREC es una versión de sólo lectura de CREA. Ofrece recursos de inspección sin proporcionar la capacidad de realizar cambios.

Para obtener descripciones completas de CREA y CREC, consulte CREA - create REQUESTMODELs for enterprise beans en *Transacciones suministradas de CICS*.

Puede utilizar CREA y CREC sin necesidad de acceder a un terminal 3270. Para obtener detalles, consulteConnecting CICS to the web en *Guía de Internet de CICS*.

Utilización de herramientas de despliegue de CICS para enterprise beans

Para desarrollar y desplegar un bean en CICS, un desarrollador de aplicaciones, en colaboración con un programador del sistema CICS en fases posteriores, debe lleva a cabo distintos pasos.

Acerca de esta tarea

Desarrolle el bean y hágalo desplegable

Desarrolle el bean y empaquételo en un archivo JAR. El bean se puede escribir y probar utilizando las herramientas que prefiera.

Nota: El archivo JAR puede contener las clases Java para uno o varios enterprise beans. Generalmente, un archivo JAR utilizado en un servidor EJB de CICS contiene varios enterprise beans.
Tras haber empaquetado el bean en un archivo JAR, utilice ATK para hacerlo desplegable. Para obtener una breve introducción a ATK y una referencia para obtener más información, consulte The enterprise bean deployment tool, ATK, en la *Guía de operaciones y programas de utilidad de CICS*.

Almacénelo en el directorio de recogida de z/OS UNIX

Almacene una copia del archivo JAR desplegable en el directorio de recogida de z/OS UNIX del CorbaServer en el que desea ejecutar el bean. Puede hacerlo mediante FTP, NFS o SMB. Si el directorio de z/OS UNIX se puede montar en su estación de trabajo, este proceso se puede integrar en el proceso de creación del archivo JAR anterior.

Explore el directorio de recogida

Mediante CEMT o el Gestor de recursos para enterprise beans, inicie una exploración del directorio de recogida. (Si desea una descripción del Gestor de recursos para enterprise beans, consulte The Resource Manager for Enterprise Beans, en la *Guía de operaciones y programas de utilidad de CICS*). CICS crea e instala una definición DJAR para el archivo JAR desplegado en el directorio de recogida.

Tras haber explorado el directorio de recogida, puede visualizar el estado de la nueva definición DJAR para determinar si el archivo JAR desplegado está listo para su uso.

Si el archivo JAR desplegado no está listo para su uso, un desarrollador de aplicaciones puede determinar y, en la mayoría de los casos, corregir la causa del error sin necesidad de implicar a un programador del sistema.

Publicación

Publique una referencia a la interfaz inicial de cada bean del archivo JAR desplegado en un espacio de nombres externo. Los clientes pueden acceder al espacio de nombres mediante JNDI.

Si se especifica AUTOPUBLISH(YES) en la definición de CORBASERVER, los beans de un archivo JAR desplegado se publican automáticamente en el espacio de nombres cuando la definición del DJAR se instala correctamente en el CorbaServer. Como alternativa, puede emitir un mandato PERFORM CORBASERVER PUBLISH o PERFORM DJAR PUBLISH.

El Gestor de recursos para enterprise beans (consulte The Resource Manager for Enterprise Beans, en la *Guía de operaciones y programas de utilidad de CICS*) indica si la función de "autopublicar" está activada o desactivada.

Asegúrese de que cualquier clase adicional está en una vía de acceso de clases
En el caso de enterprise beans, no es necesario añadir los archivos JAR desplegados a las vías de acceso de clases del perfil de JVM o del archivo de propiedades de JVM. CICS gestiona la carga de las clases incluidas en estos archivos mediante las definiciones de DJAR. Sin embargo, si los enterprise beans utilizan cualquier clase, como clases para programas de utilidad, que **no** esté incluida en el archivo JAR desplegado, tiene que incluir dichas clases en la vía de acceso de clases que utilizará la JVM para el programa procesador de solicitudes. En Capítulo 5, "Habilitación de aplicaciones para que utilicen una JVM", en la página 87 se explica cómo realizar esta tarea.

Prueba de unidad
Una vez que se hayan publicado los beans del archivo JAR desplegado en el servidor de nombres, el programador de aplicaciones puede realizar una prueba de unidad de los mismos en el entorno CICS.

Prueba de sistema
Cuando los beans estén listos para las pruebas de sistema, un programador de aplicaciones puede trabajar junto a un programador del sistema para considerar si es necesaria alguna definición REQUESTMODEL. Utilice la transacción CREA proporcionada por CICS para generar definiciones REQUESTMODEL. (Para obtener una descripción de CREA, consulte CREA - create REQUESTMODELs for enterprise beans, en el manual *Transacciones suministradas de CICS*).

Puede identificar los beans y los métodos de bean desde la aplicación. El programador del sistema puede asociar los métodos de bean con ID de transacción haciendo que se genere el conjunto óptimo de definiciones REQUESTMODEL. Ejecutar distintos beans con distintos ID de transacción resulta útil, por ejemplo, para gestión de carga de trabajo y para recopilación de información efectiva de supervisión y estadística.

Instalación en el entorno de producción
Para mover de una prueba de producción a un entorno de producción:

1. Utilice ATK para verificar que los enlaces de contenedor para recursos y referencias que se han definido en el descriptor de despliegue de cada archivo JAR son adecuados para su entorno de producción.
2. Si ha definido el parámetro DJARDIR en la definición de CORBASERVER de su región de producción para identificar un directorio de recogida:
   a. Almacene el archivo JAR desplegable en el directorio de recogida del CorbaServer.
   b. Instale la definición CORBASERVER.
   c. Se produce una definición DJAR adecuada.
3. Si no es así:
   a. Almacene el archivo JAR desplegable en el directorio de recogida de z/OS UNIX que pretende utilizar en la región de producción.
   b. Instale la definición CORBASERVER de producción.
   c. Cree e instale una definición DJAR equivalente a la que tenía en la región de prueba utilizando aquellos procesos que utilizaría normalmente en su instalación.
4. Si ha definido el parámetro AUTOPUBLISH(YES) en la definición de CORBASERVER de su región de producción:

a. Los beans del archivo JAR desplegado se publican automáticamente en el espacio de nombres cuando la definición de DJAR se instala adecuadamente en el CorbaServer.

5. Si no es así:
   a. Publique los beans en el servidor de JNDI que utilice para producción mediante CEMT PERFORM CORBASERVER PUBLISH o CEMT PERFORM DJAR PUBLISH.
6. Transfiera las definiciones REQUESTMODEL desde el CSD de la región de prueba al CSD de producción mediante el proceso que utilizaría normalmente en su instalación.
7. Asegúrese de que cualquier clase adicional, como clases para programas de utilidad, que no esté incluida en los archivos JAR desplegados para sus enterprise beans, esté presente en la vía de acceso de clase estándar.

Nota: Si desea actualizar enterprise beans en una región de producción, consulte "Actualización de enterprise beans en una región de producción" en la página 326.

Ajuste para enterprise beans

Si utiliza enterprise beans en su sistema CICS, esta información sobre ajustes puede resultarle de ayuda:

Acerca de esta tarea

- Un uso intensivo de enterprise beans puede implicar que sea necesario aumentar el tamaño del almacén de objetos EJB, DFHEJOS. En "Personalización de DFHEJOS para el uso previsto de enterprise beans con estado" se explica cómo hacerlo.
- La utilización de transacciones del OTS (servicio de transacciones de objetos) controladas por el cliente puede afectar a los requisitos para las JVM. En "Enterprise beans implicados en transacciones del OTS (servicio de transacciones de objetos) controlado por el cliente" en la página 325 se explica qué buscar.
- La utilización de más de un procesador de solicitudes por un único método de enterprise bean puede dar origen a puntos muertos. En "Métodos de enterprise bean que requieren varios procesadores de solicitudes" en la página 325 se explica cómo eliminar esta posibilidad.

Personalización de DFHEJOS para el uso previsto de enterprise beans con estado

El almacén de objetos EJB, DFHEJOS, es un archivo utilizado para almacenar beans de sesión con estado que se hayan desactivado. Puede ser un archivo VSAM o una tabla de datos de recurso de acoplamiento. CICS proporciona JCL de ejemplo para ayudarle a crear este archivo, en el miembro DFHDEFDS de la biblioteca SDFHINST.

Los valores proporcionados por CICS para DFHEJOS están diseñados para el almacenamiento de un número pequeño de objetos (beans desactivados) con un tamaño máximo de 8 K, para minimizar el malgasto de almacenamiento. Si tiene previsto un gran uso de enterprise beans con estado, aumente las asignaciones de espacio y los tamaños de registro para este conjunto de datos.

En Defining the EJB data sets en la *Guía de definición del sistema CICS* se describe cómo crear DFHEJOS y el procedimiento para calcular los valores adecuados para los tamaños de registro.

Enterprise beans implicados en transacciones del OTS (servicio de transacciones de objetos) controlado por el cliente

El uso de transacciones del OTS (servicio de transacciones de objetos) controlado por el cliente puede afectar a los requisitos de su JVM.

La carga de trabajo normal de enterprise beans en CICS comienza con un mensaje de IIOP entrante, el cuál contiene una solicitud de GIOP que es recibida por una tarea de escucha de IIOP en CICS. La solicitud se pasa a una tarea de receptor de peticiones que examina el mensaje de GIOP y pasa el proceso del mensaje a una tarea del procesador de solicitudes. Finalmente, a la terminación de la tarea del procesador de solicitudes, se envía una respuesta al cliente solicitante mediante la tarea del receptor de peticiones.

Si la solicitud de GIOP forma parte de una transacción del OTS controlado por el cliente, las tareas del procesador de solicitudes y del receptor de peticiones no finalizan hasta que la transacción del OTS se confirma o se retrotrae. Como la tarea del procesador de solicitudes se ejecuta en una JVM, esa JVM no está disponible para que la utilice ninguna otra tarea hasta que haya finalizado la transacción del OTS. Si esto sucede con frecuencia, tal vez sea necesario aumentar el número de JVM en la agrupación de JVM para evitar tiempos de espera excesivos para solicitudes entrantes.

Métodos de enterprise bean que requieren varios procesadores de solicitudes

Acerca de esta tarea

Si una única ejecución de un método de enterprise bean necesita más de un procesador de solicitudes, su aplicación podría sufrir problemas de punto muerto. (Se puede decir que un método "necesita más de un procesador de solicitudes" si llama a uno o más métodos distintos, generalmente remotos, cada uno de los cuales debe ejecutarse en un procesador de solicitudes distinto). Los puntos muertos pueden deberse a que todos los procesadores de solicitudes necesarios para satisfacer el método se vean forzados a esperar a una JVM cuando no se permitan más JVM. Esto puede deberse a dos razones:

1. En este caso sencillo, el número máximo de JVM permitido que exista simultáneamente en CICS (MAXJVMTCBS) es menor que el número de procesadores de solicitudes para atender la solicitud de método.
2. En el caso complejo:
   - CICS procesa varias solicitudes a la vez.
   - Todas las solicitudes esperan otra JVM.
   - Todas las JVM permitidas se están utilizando actualmente.

Evitar el caso sencillo es fácil, evitar el complejo es más difícil. Es necesario asegurarse siempre de que hay suficientes JVM libres para permitir satisfacer al menos la necesidad de instancias de procesador de solicitudes de un método.

El número máximo de JVM simultáneas disponible para un método de bean se define mediante el atributo MAXACTIVE de la definición TRANCLASS que se aplica a la transacción del procesador de solicitudes. El número máximo de JVM simultáneas disponible para CICS se define mediante el parámetro de inicialización del sistema MAXJVMTCBS.

Para eliminar la posibilidad de puntos muertos causados por métodos de bean que utilizan varios procesadores de solicitudes:

1. Siempre que sea coherente con los requisitos de su aplicación, trate de minimizar el número de procesadores de solicitud que necesita cada método, preferiblemente a uno. Si puede reducir los requisitos de todos los métodos, en todas las aplicaciones, a un único procesador de solicitudes, no tiene que hacer más.
2. Si no es posible reducir los requisitos de todos los métodos a un único procesador de solicitudes, descubra cuál es su "peor caso": es decir, el método de bean que requiere más procesadores de solicitudes para completarlo correctamente.
3. Cree una nueva definición TRANCLASS. Esta clase de transacción se aplicará a la transacción del procesador de solicitudes en el que se ejecutarán los métodos de bean que necesitan varios procesadores de solicitudes.
4. En la definición de TRANCLASS, defina el valor de MAXACTIVE mediante la siguiente fórmula:

   ```
   MAXACTIVE <= ((MAXJVMTCBS - n) / (n - 1)) + 1
   ```

   donde `n` es el número máximo de procesadores de solicitudes necesarios para su método de "peor caso".

   Si el resultado de este cálculo es un valor decimal, redondéelo al número entero más cercano (inferior).
5. Cree nuevas definiciones TRANSACTION y REQUESTMODEL:
   a. Cree una nueva definición TRANSACTION para la transacción del procesador de solicitudes en el que se ejecutarán los métodos de bean que necesitan varios procesadores de solicitudes. (El modo más fácil para hacerlo es copiar la definición de la transacción predeterminada CIRP del procesador de solicitudes y modificarla). En la opción TRANCLASS, especifique el nombre de la nueva clase de transacción.
   b. Cree una o más definiciones REQUESTMODEL. Entre ellas, sus nuevas definiciones REQUESTMODEL deben cubrir todas las solicitudes que pueden recibirse para métodos de bean que necesitan varios procesadores de solicitudes. En la opción TRANSID de las definiciones REQUESTMODEL, especifique el nombre de la nueva transacción.

Actualización de enterprise beans en una región de producción

Esta sección analiza el mejor modo de actualizar enterprise beans en una región de producción. Contiene los temas siguientes:

- "El problema"
- "Soluciones posibles" en la página 328

El problema

¿Cómo se actualizan enterprise beans en una región de producción CICS en ejecución provocando las menores alteraciones posibles en el flujo de trabajo actual y sin reiniciar CICS?

Es muy sencillo introducir nuevos enterprise beans en un servidor EJB en ejecución sin alterar el flujo de trabajo actual. Puede hacer cualquiera de las siguientes cosas:

1. Utilice el mecanismo de exploración de CICS. Es decir, coloque el archivo JAR desplegado que contiene los nuevos beans en el directorio de recogida del archivo JAR desplegado de un CorbaServer y emitir un mandato **PERFORM CORBASERVER SCAN**. Repita en todas las AOR del servidor EJB lógico. Si la definición CORBASERVER especifica AUTOPUBLISH(NO), en una de las AOR emita un mandato **PERFORM DJAR PUBLISH**.

Nota: Si utiliza el mecanismo de exploración en una región de producción, tenga en cuenta las implicaciones de seguridad: específicamente, la posibilidad de que se omita la seguridad de mandatos de CICS en definiciones DJAR. Para protegerse contra esto, recomendamos que los ID de usuario a los que se otorgue acceso de grabación al archivo JAR desplegado z/OS UNIX se restrinjan a aquellos a los que se haya otorgado autoridad RACF para crear y actualizar definiciones DJAR y CORBASERVER.

2. Utilice un mandato **EXEC CICS CREATE DJAR** para instalar una definición del archivo JAR desplegado que contenga los nuevo beans. Repita en todas las AOR del servidor EJB lógico. En una de las AOR, emita un mandato **PERFORM DJAR PUBLISH**.

Por desgracia, debido a los efectos imprevisibles sobre las transacciones en curso, no puede utilizar estos métodos para *actualizar* beans en un servidor EJB activo. No tendría forma de controlar qué versión de un bean, la antigua o la nueva, utilizaron llamadas a método sucesivas. (Debido a las diferencias de temporización, el problema podría aumentar en un servidor EJB multirregión).

Un método alternativo sería desactivar temporalmente y apagar CICS, y luego reiniciarlo con las definiciones DJAR actualizadas instauradas. Aunque esto es aceptable en un entorno de prueba, no resulta una solución atractiva para una región de producción. Piense en la Figura 24 en la página 328. Imagine que desea actualizar el `bean5` y el `bean6` en el CorbaServer `COR2`. Si fuera a cerrar CICS, no solo el `bean5` y el `bean6` no estarían disponibles durante la conclusión, sino que tampoco lo estarían todos los beans del CorbaServer `COR1`.

¿Qué sucede si el servidor EJB contiene varias AOR y se utiliza la gestión de carga de trabajo para equilibrar las solicitudes en ellas? ¿No se podría entonces apagar y actualizar cada AOR por turno con un efecto mínimo sobre el rendimiento? Por desgracia, no, ya que:

- Durante el proceso de actualización, distintas AOR tendrían distintas versiones de los beans. A no ser que las nuevas versiones de los beans fueran totalmente compatibles con las versiones antiguas, esto provocaría efectos imprevisibles. (Para que la nueva versión de un bean sea totalmente compatible con una versión antigua, entre otras cosas, las interfaces inicial y de componente de ambas versiones deben ser idénticas, y el estado de cualquier bean de sesión con estado debe conservarse).
- Apagar incluso una sola AOR inevitablemente degradaría el rendimiento del servidor EJB hasta cierto punto. (Si la actualización es importante, esto puede ser aceptable. Para compensar la reducción de rendimiento, quizá podría añadir una AOR extra al servidor EJB).

El resto de esta sección trata sobre lo que hay que hacer en un servidor EJB de CICS para actualizar enterprise beans en regiones de producción. Tenga en cuenta que también pueden ser necesarios cambios en el lado del cliente. En concreto, si debido a una actualización cambia la interfaz inicial o de componente de un enterprise bean, antes de que cualquier aplicación cliente pueda utilizar el bean actualizado, deben volver a escribirse para que utilicen la nueva interfaz.

La siguiente figura muestra los clientes que invocan métodos de bean en los CorbaServers COR1 y COR2. Puede dividir los beans entre CorbaServers en base a los requisitos de mantenimiento y disponibilidad de los beans.

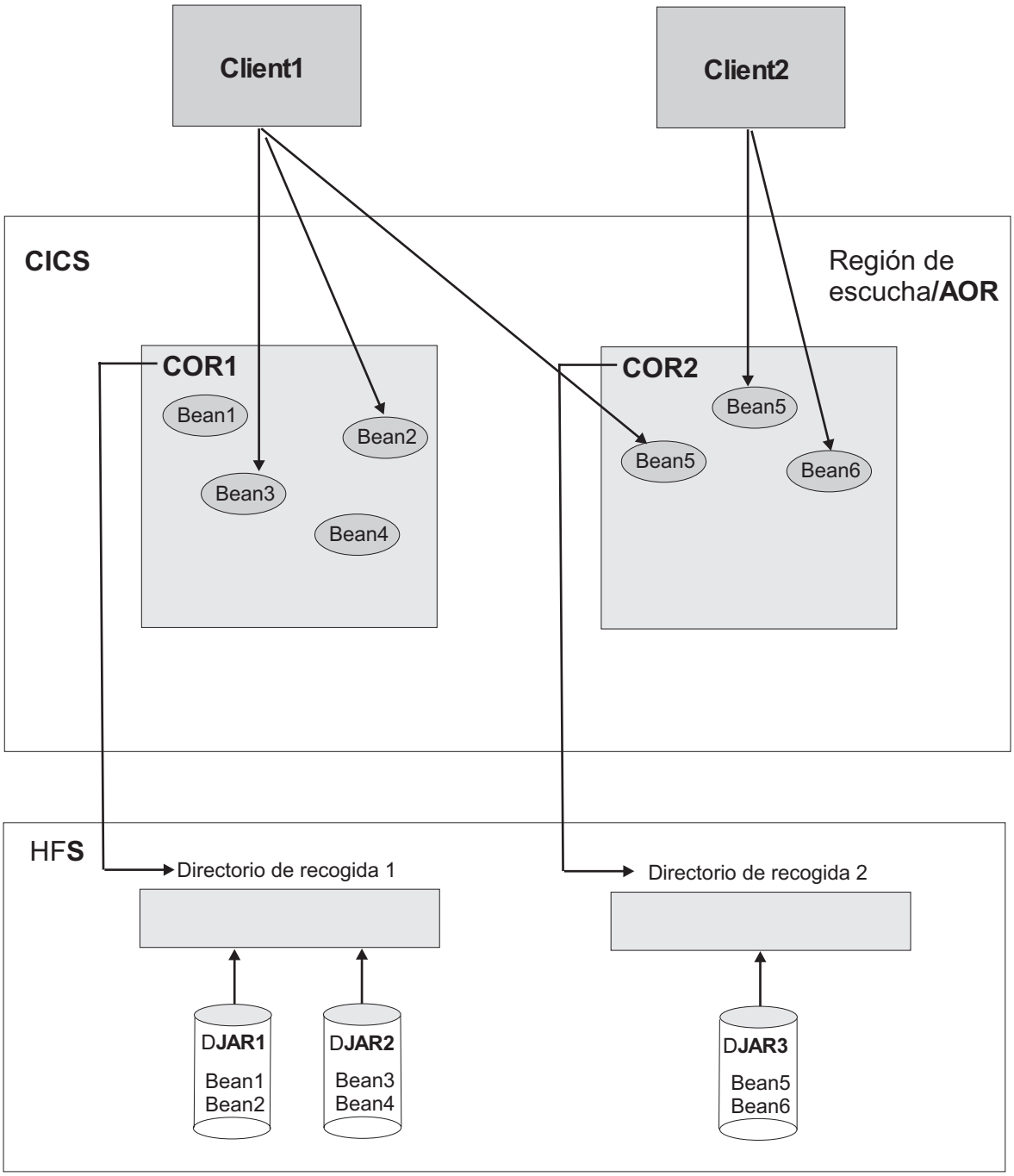


Figura 24. Una región de producción EJB de CICS.

Para obtener algunas soluciones sugeridas para el problema de cómo actualizar mejor los beans en una región de producción, consulte "Soluciones posibles".

Soluciones posibles

Aquí le presentamos algunas soluciones sugeridas para nuestro problema de cómo actualizar mejor los beans en una región de producción. Las soluciones ofrecidas dependen de si el servidor EJB consta de una única región de escucha/AOR o de varias regiones de escucha y varias AOR.

Como regla general, las soluciones de actualización serán más fáciles de implementar si:

1. Divide sus enterprise beans entre CorbaServers en base no solo a las funciones de los beans, sino también a sus requisitos de mantenimiento y disponibilidad. Es decir, conjuntos de beans que tienen distintos requisitos de mantenimiento y disponibilidad deben instalarse en CorbaServers distintos.
2. Asigne ID de transacción de CICS a métodos de enterprise bean en base no solo a las funciones de los beans, sino también a sus requisitos de mantenimiento y disponibilidad. Es decir, para facilitar el mantenimiento, conjuntos de beans que tienen distintos requisitos de mantenimiento y disponibilidad deben ejecutarse con ID de transacción de CICS distintos.

   Importante:

   a. En un servidor EJB multirregión, si las AOR contienen varios CorbaServers, se recomienda encarecidamente asignar distintos conjuntos de ID de transacción a los objetos soportados por cada CorbaServer. Es decir, cada CorbaServer de una AOR debe soportar un conjunto distinto de ID de transacción.
   b. Esto hace que sea más fácil que el programa de direccionamiento evite un CorbaServer inhabilitado mientras se mantienen disponibles los demás CorbaServers habilitados de la región. Para obtener más información sobre cómo codificar un programa de direccionamiento distribuido para gestionar un CorbaServer inhabilitado, consulte la *Guía de personalización de CICS*.

   Nota: La transacción de CICS en la que se ejecuta un método de bean se especifica en la definición de REQUESTMODEL que coincide con el método. Puede utilizar la transacción CREA provista por CICS para lo siguiente:
   - Visualice los ID de transacción asociados con beans y métodos de bean concretos.
   - Cambie los ID de transacción, aplique los cambios y guarde dichos cambios en las nuevas definiciones de REQUESTMODEL.

Soluciones para una región de escucha/AOR única:

Estas soluciones son válidas para un servidor EJB que conste de una única región de escucha/AOR.

Asumamos que, en la Figura 24 en la página 328, desea actualizar el `bean5` y el `bean6` en el CorbaServer `COR2`. `DJAR3.jar` es el archivo JAR desplegado que contiene los beans que actualizar. Es necesario que:

1. El CorbaServer `COR1` y sus beans estén disponibles durante el proceso de actualización.
2. Si es posible, la actualización de los beans en el CorbaServer `COR2` se produzca sin interrupciones. Es decir, no debería de haber ningún momento (o, al menos, el periodo de tiempo debería de ser el mínimo posible) durante el que sea imposible crear una nueva instancia de `bean5` o de `bean6`.

Solución 1:
Acerca de esta tarea

La ventaja de esta solución es que resulta relativamente fácil de implementar. La desventaja es que no es directa: es decir, hay un periodo (mientras las versiones antiguas de `bean5` o de `bean6` se destruyen o desactivan) durante el que resulta imposible crear una nueva instancia de `bean5` o de `bean6`.

1. Emita un mandato **EXEC CICS SET CORBASERVER**(COR2) ENABLESTATUS(DISABLED) o un mandato **CEMT SET CORBASERVER**(COR2) DISABLED. Cualquier intento de crear una nueva instancia de bean5 o bean6, independientemente de si los clientes tienen referencias a las interfaces iniciales de los beans, fracasará.

   Generalmente, los método actualmente en ejecución en instancias de bean5 y bean6 seguirán hasta su terminación.

   Una instancia de bean5 o bean6 que no participe en una transacción del OTS se destruye o se desactiva al final del método actualmente en ejecución. (Si no hay ningún método en ejecución actualmente, todas las instancias ya se habrán destruido o desactivado).

   Nota: Los beans de sesión *sin estado* se destruyen. Los beans de sesión *con estado* se desactivan.

   Una instancia de bean5 o bean6 que *sí* participe en una transacción del OTS no se destruye ni se desactiva hasta el final de la transacción del OTS; generalmente cualquier llamada a método futura para esta instancia (en el ámbito de la transacción del OTS) se realizará correctamente. Al final de la transacción del OTS, la instancia se destruye o se desactiva.
2. Compruebe cuándo se han destruido o desactivado todas las instancias de bean5 y bean6 emitiendo los mandatos **EXEC CICS** o **CEMT INQUIRE CORBASERVER**(COR2) ENABLESTATUS. Un estado de DISABLED (inhabilitada) indica que todas las instancias de bean se han destruido o desactivado.
3. Cuando todas las instancias de bean5 y bean6 se hayan destruido o desactivado, instale la versión actualizada del archivo JAR desplegado DJAR3.jar, mediante el mecanismo de exploración de CICS o mediante una definición DJAR estática. (No puede utilizar el mecanismo de exploración para actualizar una definición DJAR estática).

   Puede hacer lo siguiente:

   a. Ponga la nueva versión del archivo JAR desplegado DJAR3.jar en el directorio de recogida de COR2 de CorbaServer.
   b. Emita un mandato **PERFORM CORBASERVER**(COR2) SCAN. CICS explora el directorio de recogida de COR2, instala la nueva definición de DJAR3.jar y copia las nuevas versiones de bean5 y de bean6 en el directorio de estantería de COR2.

   O bien:

   a. Emita un mandato **EXEC CICS** o **CEMT DISCARD DJAR**(DJAR3) para eliminar la definición actual de DJAR3.jar de CICS.
   b. Emita un mandato **CEDA INSTALL DJAR**(DJAR3) o un mandato **EXEC CICS CREATE DJAR**(DJAR3) CORBASERVER (COR2) HFSFILE(nueva_versión_de_DJAR3.jar_en_HFS). CICS instala la nueva definición de DJAR3.jar y copia las nuevas versiones de bean5 y de bean6 en el directorio de estantería de COR2.

   Nota:

   a. *No* es necesario volver a publicar las versiones actualizadas de bean5 y de bean6 en el espacio de nombres, incluso si las interfaces inicial y de componente de los beans se han cambiado desde la versión anterior.
   b. Si la interfaz inicial o de componente de bean5 o de bean6 se ha cambiado desde la versión anterior, es necesario actualizar las aplicaciones cliente del bean modificadas para que utilicen la nueva firma antes de utilizarlas.

c. Si actualiza un bean de sesión *con estado*, en función de exactamente qué cambios se realicen, puede cambiar la estructura de su estado serializado. Si esto sucede, invalidará cualquier instancia desactivada del bean en el almacén de objetos. En este caso, cualquier intento de utilizar el bean que ahora está invalidado resultará en una excepción. Debe codificar sus aplicaciones cliente para que hagan frente a esta posibilidad.

4. Emita un mandato **CEMT SET CORBASERVER**(COR2) ENABLED. *Desde este momento, todo el trabajo nuevo utilizará las versiones actualizadas de bean5 y de bean6.*

Solución 2:

Esta solución requiere CICSPlex System Manager. Todas las aplicaciones CICS en la región de escucha/AOR deben ser adecuadas para clonación en varias regiones.

Acerca de esta tarea

La ventaja de esta solución es que, a diferencia de la solución 1, es relativamente directa: es decir, en el peor de los caso, debe de haber solo un pequeño periodo durante el que resulte imposible crear una nueva instancia de bean5 o de bean6. La desventaja es que resulta más complicada de implementar que la solución 1.

1. Mediante CICSPlex SM:
   a. Clone su región de escucha/AOR única.
   b. Dirija toda la carga de trabajo nueva al clon: es decir, desactive temporalmente la AOR original y active el clon. Para obtener información sobre cómo hacerlo, consulte Balancing an enterprise bean workload, en el manual *Cargas de trabajo de gestión de CICSPlex System Manager*.

      Todas las solicitudes de métodos de bean que se ejecutarán en una nueva transacción del OTS o sin ninguna transacción del OTS, tanto en COR1 como en COR2, ahora se dirigen al clon.

      Las solicitudes de métodos de bean que se ejecutarán en una transacción del OTS existente, tanto en COR1 como en COR2, se dirigen a la región original.

      Nota:

      1) Por "una *nueva* transacción del OTS" nos referimos a una transacción del OTS en la que la participación del bean comience *después* de que todo el trabajo nuevo se dirija al clon.
      2) Por "una transacción del OTS *existente*" nos referimos a una transacción del OTS en la que la participación del bean comenzara *antes* de que todo el trabajo nuevo se dirigiera al clon.

      En la región original:

      - Una instancia de un enterprise bean que no participe en una transacción del OTS se destruye o se desactiva al final del método actualmente en ejecución. (Si no hay ningún método en ejecución actualmente, todas las instancias ya se habrán destruido o desactivado).
      - Una instancia de un enterprise bean que *sí* participe en una transacción del OTS no se destruye ni se desactiva hasta el final de la transacción del OTS; generalmente cualquier llamada a método futura para esta instancia (en el ámbito de la transacción del OTS) se realizará correctamente. Al final de la transacción del OTS, la instancia se destruye o se desactiva.

2. En la región original:
   a. Compruebe cuándo se han destruido o desactivado todas las instancias desde bean1 a bean6:

1) Si aún no conoce el ID o los ID de transacción de CICS asociados con bean1 a bean6, utilice la transacción CREC para visualizar esta información.
2) Utilice el mandato **INQUIRE TASK** para comprobar si se está ejecutando cualquier instancia de estas transacciones.

b. Cuando todas las instancias de bean1 a bean6 se hayan destruido o desactivado, instale la versión actualizada del archivo JAR desplegado DJAR3.jar, mediante el mecanismo de exploración de CICS o mediante una definición DJAR estática. (No puede utilizar el mecanismo de exploración para actualizar una definición DJAR estática).

Puede hacer lo siguiente:

1) Ponga la nueva versión del archivo JAR desplegado DJAR3.jar en el directorio de recogida de COR2 de CorbaServer.
2) Emita un mandato **PERFORM CORBASERVER**(COR2) SCAN. CICS explora el directorio de recogida de COR2, actualiza su definición de DJAR3.jar y copia las nuevas versiones de bean5 y de bean6 en el directorio de estantería de COR2.

O bien:

1) Emita un mandato **CEMT DISCARD DJAR**(DJAR3) para suprimir la definición antigua de DJAR3.jar.
2) Emita un mandato **CEDA INSTALL DJAR**(DJAR3) o un mandato **EXEC CICS CREATE DJAR**(DJAR3) CORBASERVER (COR2) HFSFILE(nueva_versión_de_DJAR3.jar_en_HFS). CICS instala la nueva definición de DJAR3.jar y copia las nuevas versiones de bean5 y de bean6 en el directorio de estantería de COR2.

Nota:

1) *No* es necesario volver a publicar las versiones actualizadas de bean5 y de bean6 en el espacio de nombres, incluso si las interfaces inicial y de componente de los beans se han cambiado desde la versión anterior.
2) Si la interfaz inicial o de componente de bean5 o de bean6 se ha cambiado desde la versión anterior, es necesario actualizar las aplicaciones cliente del bean modificadas para que utilicen la nueva firma antes de utilizarlas.
3) Si actualiza un bean de sesión *con estado*, en función de exactamente qué cambios se realicen, puede cambiar la estructura de su estado serializado. Si esto sucede, invalidará cualquier instancia desactivada del bean en el almacén de objetos. En este caso, cualquier intento de utilizar el bean que ahora está invalidado resultará en una excepción. Debe codificar sus aplicaciones cliente para que hagan frente a esta posibilidad.

3. Mediante CICSPlex SM, dirija toda la carga de trabajo nueva a la región original: es decir, desactive temporalmente el clon y active la región original.

Todas las solicitudes de métodos de bean que se ejecutarán en una nueva transacción del OTS o sin ninguna transacción del OTS, tanto en COR1 como en COR2, ahora se dirigen a la región original. *Desde este momento, todo el trabajo nuevo utilizará las versiones actualizadas de bean5 y de bean6*. Las solicitudes de métodos de bean que se ejecutarán en una transacción del OTS existente, tanto en COR1 como en COR2, se siguen dirigiendo al clon.

Nota:

a. Por "una *nueva* transacción del OTS" nos referimos a una transacción del OTS en la que la participación del bean comience *después* de que todo el trabajo nuevo se redirija a la región original.
b. Por "una transacción del OTS *existente*" nos referimos a una transacción del OTS en la que la participación del bean comenzara *antes* de que todo el trabajo nuevo se redirigiera a la región original.

Finalmente, todas las instancias de enterprise beans en el clon se destruirán o se desactivarán, como se describe anteriormente.

4. En la región clonada, utilice el mandato **INQUIRE TASK** para comprobar cuándo se han destruido o desactivado todas las instancias desde `bean1` a `bean6`. Cuando esto haya sucedido, puede descartar la región clonada.

Soluciones para un servidor EJB multirregión:

Estas soluciones son válidas para un servidor EJB que conste de una o más regiones de escucha y de varias AOR idénticas.

Asumamos que el servidor EJB consta de tres regiones de escucha idénticas y de cinco AOR idénticas. Cada una de las AOR es un clon de la región que se muestra en la Figura 24 en la página 328 (excepto que se trata de una AOR en lugar de una región de escucha/AOR). Todas las AOR comparten los mismos directorios de recogida y se despliegan los mismos conjuntos de enterprise beans en cada una, en CorbaServers idénticos llamados `COR1` y `COR2`.

Desea actualizar el `bean5` y el `bean6` en el CorbaServer lógico `COR2`. `DJAR3.jar` es el archivo JAR desplegado que contiene los beans que actualizar.

Es necesario que:

1. El CorbaServer lógico `COR1` y sus beans estén disponibles durante el proceso de actualización.
2. Si es posible, la actualización de los beans en el CorbaServer lógico `COR2` se produzca sin interrupciones. Es decir, no debería de haber ningún momento (o, al menos, el periodo de tiempo debería de ser el mínimo posible) durante el que sea imposible crear una nueva instancia de `bean5` o de `bean6`.

Solución 1:

Esta solución es un desarrollo de la solución 1 para una región única.

Acerca de esta tarea

Su ventaja es que resulta relativamente fácil de implementar. Su desventaja es que no es directa: es decir, hay un periodo (mientras las versiones antiguas de `bean5` o de `bean6` se destruyen o se desactivan) durante el que resulta imposible crear una nueva instancia de `bean5` o de `bean6`.

1. En cada una de las AOR, emita un mandato **EXEC CICS SET CORBASERVER**(`COR2`) ENABLESTATUS(`DISABLED`) o un mandato **CEMT SET CORBASERVER(**`COR2`) DISABLED. En todas las AOR:
   - Cualquier intento de crear una nueva instancia de `bean5` o `bean6`, independientemente de si los clientes tienen referencias a las interfaces iniciales de los beans, fracasará.
   - Generalmente, los método actualmente en ejecución en instancias de `bean5` y `bean6` seguirán hasta su terminación.

- Una instancia de bean5 o bean6 que no participe en una transacción del OTS se destruye o se desactiva al final del método actualmente en ejecución. (Si no hay ningún método en ejecución actualmente, todas las instancias ya se habrán destruido o desactivado).
- Una instancia de bean5 o bean6 que *sí* participe en una transacción del OTS no se destruye ni se desactiva hasta el final de la transacción del OTS; generalmente cualquier llamada a método futura para esta instancia (en el ámbito de la transacción del OTS) se realizará correctamente. Al final de la transacción del OTS, la instancia se destruye o se desactiva.

2. En cada una de las AOR, compruebe cuándo se han destruido o desactivado todas las instancias de bean5 y bean6 emitiendo los mandatos **EXEC CICS** o **CEMT INQUIRE CORBASERVER**(COR2) ENABLESTATUS. Un estado de DISABLED (inhabilitada) indica que todas las instancias de bean se han destruido o desactivado.
3. Cuando todas las instancias de bean5 y bean6, en todas las AOR originales, se hayan destruido o desactivado, instale la versión actualizada del archivo JAR desplegado DJAR3.jar, mediante el mecanismo de exploración de CICS o mediante las definiciones DJAR estáticas. (No puede utilizar el mecanismo de exploración para actualizar definiciones DJAR estáticas).

   Puede hacer lo siguiente:

   a. Ponga la nueva versión del archivo JAR desplegado DJAR3.jar en el directorio de recogida de COR2 de CorbaServer (que comparten todas las AOR).
   b. En cada una de las AOR, emita un mandato **PERFORM CORBASERVER**(COR2) SCAN. La AOR explora el directorio de recogida de COR2, instala la nueva definición de DJAR3.jar y copia las nuevas versiones de bean5 y de bean6 en el directorio de estantería de COR2.

   O bien, en cada una de las AOR:

   a. Emita un mandato **EXEC CICS** o **CEMT DISCARD DJAR**(DJAR3) para eliminar la definición actual de DJAR3.jar de CICS.
   b. Emita un mandato **CEDA INSTALL DJAR**(DJAR3) o un mandato **EXEC CICS CREATE DJAR**(DJAR3) CORBASERVER (COR2) HFSFILE(nueva_versión_de_DJAR3.jar_en_HFS). CICS instala la nueva definición de DJAR3.jar y copia las nuevas versiones de bean5 y de bean6 en el directorio de estantería de COR2.

   Nota:

   a. *No* es necesario volver a publicar las versiones actualizadas de bean5 y de bean6 en el espacio de nombres, incluso si las interfaces inicial y de componente de los beans se han cambiado desde la versión anterior.
   b. Si la interfaz inicial o de componente de bean5 o de bean6 se ha cambiado desde la versión anterior, es necesario actualizar las aplicaciones cliente del bean modificadas para que utilicen la nueva firma antes de utilizarlas.
   c. Si actualiza un bean de sesión *con estado*, en función de exactamente qué cambios se realicen, puede cambiar la estructura de su estado serializado. Si esto sucede, invalidará cualquier instancia desactivada del bean en el almacén de objetos. En este caso, cualquier intento de utilizar el bean que ahora está invalidado resultará en una excepción. Debe codificar sus aplicaciones cliente para que hagan frente a esta posibilidad.
4. En cada una de las AOR, emita un mandato **CEMT CORBASERVER**(COR2) ENABLED. *Desde este momento, todo el trabajo nuevo utilizará las versiones actualizadas de bean5 y de bean6*.

Solución 2:
Acerca de esta tarea

Esta solución requiere CICSPlex System Manager. Es un desarrollo de la solución 2 para una región única. Su ventaja es que es relativamente directa: es decir, en el peor de los casos, debe de haber solo un pequeño periodo durante el que resulte imposible crear una nueva instancia de `bean5` o de `bean6`. Su desventaja es que resulta más complicada de implementar que la solución 1.

1. Mediante CICSPlex SM:
   a. Cree clones de todas sus AOR.
   b. Dirija toda la carga de trabajo nueva a los clones: es decir, desactive temporalmente las AOR originales y active los clones. Para obtener información sobre cómo hacerlo, consulte Balancing an enterprise bean workload, en el manual *Cargas de trabajo de gestión de CICSPlex System Manager*.

      Cada solicitud de un método de bean que se ejecutará en una nueva transacción del OTS o sin ninguna transacción del OTS, tanto en COR1 como en COR2, se dirige a uno u otro de los clones.

      Cada solicitud de un método de bean que se ejecutará en una transacción del OTS existente, tanto en COR1 como en COR2, se dirige a la AOR original correspondiente.

      Nota:

      1) Por "una *nueva* transacción del OTS" nos referimos a una transacción del OTS en la que la participación del bean comience *después* de que todo el trabajo nuevo se dirija a los clones.
      2) Por "una transacción del OTS *existente*" nos referimos a una transacción del OTS en la que la participación del bean comenzara *antes* de que todo el trabajo nuevo se dirigiera a los clones.
      3) Por "la AOR original *correspondiente*" nos referimos a la AOR original que contiene el procesador de solicitudes para la transacción del OTS.
2. En cada una de las AOR originales:

   Compruebe cuándo se han destruido o desactivado todas las instancias desde `bean1` a `bean6`:

   a. Si aún no conoce el ID o los ID de transacción de CICS asociados con `bean1` a `bean6`, utilice la transacción CREC para visualizar esta información.
   b. Utilice el mandato **`INQUIRE TASK`** para comprobar si se está ejecutando cualquier instancia de estas transacciones.
3. Cuando se hayan destruido o desactivado todas las instancias de `bean1` a `bean6` en todas las AOR originales, instale la versión actualizada del archivo JAR desplegado `DJAR3.jar`, mediante el mecanismo de exploración de CICS o mediante las definiciones DJAR estáticas. (No puede utilizar el mecanismo de exploración para actualizar definiciones DJAR estáticas).

   Puede hacer lo siguiente:

   a. Ponga la nueva versión del archivo JAR desplegado `DJAR3.jar` en el directorio de recogida de `COR2` (que comparten todas las AOR originales).
   b. En cada una de las AOR originales, emita un mandato **`PERFORM CORBASERVER`**(`COR2`) SCAN. La AOR explora el directorio de recogida de `COR2`, actualiza su definición de `DJAR3.jar` y copia las nuevas versiones de `bean5` y de `bean6` en el directorio de estantería de `COR2`.

   O bien:

a. En cada una de las AOR originales, emita un mandato **CEMT DISCARD DJAR**(DJAR3) para suprimir la definición antigua de DJAR3.jar.
b. En cada una de las AOR originales, emita un mandato **CEDA INSTALL DJAR**(DJAR3) o un mandato **EXEC CICS CREATE DJAR**(DJAR3) CORBASERVER (COR2) HFSFILE(nueva_versión_de_DJAR3.jar_en_HFS). CICS instala la nueva definición de DJAR3.jar y copia las nuevas versiones de bean5 y de bean6 en el directorio de estantería de COR2.

Nota:

a. *No* es necesario volver a publicar las versiones actualizadas de bean5 y de bean6 en el espacio de nombres, incluso si las interfaces inicial y de componente de los beans se han cambiado desde la versión anterior.
b. Si la interfaz inicial o de componente de bean5 o de bean6 se ha cambiado desde la versión anterior, es necesario actualizar las aplicaciones cliente del bean modificadas para que utilicen la nueva firma antes de utilizarlas.
c. Si actualiza un bean de sesión *con estado*, en función de exactamente qué cambios se realicen, puede cambiar la estructura de su estado serializado. Si esto sucede, invalidará cualquier instancia desactivada del bean en el almacén de objetos. En este caso, cualquier intento de utilizar el bean que ahora está invalidado resultará en una excepción. Debe codificar sus aplicaciones cliente para que hagan frente a esta posibilidad.

4. Mediante CICSPlex SM, dirija toda la carga de trabajo nueva a las AOR originales: es decir, desactive temporalmente los clones y active las AOR originales.

   Todas las solicitudes de métodos de bean que se ejecutarán en una nueva transacción del OTS o sin ninguna transacción del OTS, tanto en COR1 como en COR2, se dirigen ahora a las AOR originales. *Desde este momento, todo el trabajo nuevo utilizará las versiones actualizadas de bean5 y de bean6*. Las solicitudes de métodos de bean que se ejecutarán en una transacción del OTS existente, tanto en COR1 como en COR2, se siguen dirigiendo a los clones.

   Nota:

   a. Por "una *nueva* transacción del OTS" nos referimos a una transacción del OTS en la que la participación del bean comience *después* de que todo el trabajo nuevo se redirija a las AOR originales.
   b. Por "una transacción del OTS *existente*" nos referimos a una transacción del OTS en la que la participación del bean comenzara *antes* de que todo el trabajo nuevo se redirigiera a las AOR originales.

   Finalmente, todas las instancias de enterprise beans en los clones se destruirán o se desactivarán.

5. En cada uno de los clones, utilice el mandato **INQUIRE TASK** para comprobar cuándo se han destruido o desactivado todas las instancias desde bean1 a bean6. Cuando esto haya sucedido, puede descartar el clon.

Otras soluciones posibles: Las soluciones descritas en "Soluciones para una región de escucha/AOR única" en la página 329 y en "Soluciones para un servidor EJB multirregión" en la página 333 no son las únicas posibilidades. Por ejemplo, otro método es el siguiente:

1. Utilice los ID de transacción no predeterminados para los procesadores de solicitudes asociados con los beans que actualizar. (En otras palabras, separe los enterprise beans por CorbaServer y por ID de transacción del modo sugerido anteriormente).

2. Inhabilite las transacciones del procesador o ponga las transacciones en una clase de transacción y reduzca el límite de TCLASS a cero.
3. Cuando todas las instancias de los beans se hayan destruido o desactivado, instale las versiones actualizadas de los archivos JAR desplegados de una de los formas descritas para las otras soluciones.

El CCI Connector for CICS TS

El CCI Connector for CICS TS ayuda a crear componentes de servidor de Enterprise JavaBean (EJB) que aprovechan los programas CICS existentes.

Visión general del CCI Connector for CICS TS

El CCI Connector for CICS TS ayuda a compilar componentes de servidor de Enterprise JavaBean (EJB) que utilizan los programas CICS existentes.

Los conectores en segundo plano:

Frecuentemente, se pueden desarrollar nuevas aplicaciones Java de forma más rápida y fiable aprovechando los programas de CICS ya existentes (que no sean Java).

Un **conector CICS** es un componente de software que permite a una aplicación de cliente Java invocar una aplicación de CICS. Normalmente, los programas cliente Java que utilizan un conector CICS son servlets.

Durante varias versiones, CICS ha brindado sopporte para conectores CICS que permiten a un programa cliente Java *que se ejecute fuera de CICS* (por ejemplo, en Windows, en UNIX o en z/OS nativo), conectarse a un programa especificado en un servidor de CICS. El CCI Connector for CICS TS permite a un programa Java o a un enterprise bean *que se ejecute en CICS Transaction Server para z/OS* enlazar con un programa de servidor de CICS.

El CCI Connector for CICS TS implementa la **Common Client Interface (CCI)** estándar del sector definida por la J2EE Connector Architecture Specification, versión 1.0.

Nota: El CICS Connector for CICS TS, introducido en CICS TS para z/OS, versión 2.1, ya no se soporta. A diferencia del CCI Connector for CICS TS, el CICS Connector for CICS TS implementaba una interfaz de cliente no estándar patentada por IBM. Para obtener consejo sobre cómo actualizar aplicaciones existentes que utilicen el CICS Connector for CICS TS para que empleen el CCI Connector for CICS TS en su lugar, consulte “Actualización del CICS Connector for CICS TS al CCI Connector for CICS TS” en la página 354.

La Common Client Interface (CCI) o interfaz de cliente común:

Esta sección presenta una visión general de la interfaz de cliente común (CCI). La Common Client Interface es parte de la J2EE Connector Architecture.

Para obtener información definitiva sobre la interfaz, consulte la especificación J2EE Connector Architecture. Para descargar la especificación, vaya al sitio web Oracle Technology Network Java y busque *J2EE Connector architecture*.

La CCI brinda una interfaz estándar que permite a los desarrolladores comunicarse con cualquier número de sistemas de información empresarial (EIS) mediante sus adaptadores de recursos específicos utilizando un estilo de programación genérico.

La CCI se basa fundamentalmente en la interfaz de cliente utilizada por Java Database Connectivity (JDBC), y es similar en su utilización de *Connections* (Conexiones) e *Interactions* (Interacciones).

En la CCI hay dos tipos diferenciados de clase; por comodidad, las llamaremos clases de *infraestructura* y clases de *entrada/salida*.

Clases de infraestructura:

Las clases de infraestructura se utilizan para solicitar una conexión a un EIS como CICS y ejecutar mandatos en el EIS pasando entrada y recuperando salida.

Las clases de infraestructura son:

ConnectionFactory
: Se utiliza un objeto ConnectionFactory para fabricar conexiones que un componente Java pueda utilizar para comunicarse con un EIS específico. Los atributos de ConnectionFactory especifican el EIS para el que se pueden crear conexiones. Un objeto ConnectionFactory es la fábrica para un objeto Connection.

Connection
: Un objeto Connection identifica una conexión exclusiva con un servidor específico. Es la fábrica para un objeto Interaction.

Interaction
: El método execute de un objeto Interaction permite controlar una interacción con un servidor. En CICS TS, el método execute toma tres argumentos: un objeto InteractionSpec que especifica el tipo de interacción y dos objetos Record que llevan los datos de entrada y de salida.

Los componentes de J2EE utilizan las clases de infraestructura para adquirir una conexión con un EIS y enviar y recibir datos. En primer lugar, un componente J2EE obtiene un objeto ConnectionFactory para el EIS concreto al que se va a acceder; por ejemplo, CICS. (El componente puede fabricar el objeto ConnectionFactory mediante un programa o, más probablemente, buscarlo en un espacio de nombres de JNDI). Utiliza ConnectionFactory para obtener un objeto Connection. Luego utiliza el objeto **Connection** para crear uno o más objetos Interaction. Ejecuta mandatos en el EIS mediante estos objetos Interaction.

La Figura 25 muestra las clases de infraestructura de CCI utilizadas para conectar con un EIS y ejecutar un mandato.

```
ConnectionFactory cf = <Lookup from JNDI namespace>
Connection conn = cf.getConnection();
Interaction int = conn.createInteraction();
int.execute(<Input output data>);
int.close();
conn.close();
```

Figura 25. Utilización de las clases de infraestructura de CCI para conectar con un EIS y ejecutar un mandato

Clases de entrada/salida:

Utilizar las clases de infraestructura proporciona una forma genérica de acceder a un EIS mediante un adaptador de recursos de J2EE.

Sin embargo, como cada EIS tiene distintas necesidades de entrada y de salida, las interfaces CCI proporcionan un modo de que los componentes de J2EE pasen información específica de EIS a un adaptador de recursos de J2EE. Un componente J2EE utiliza los siguientes tipos de objeto para este fin:

- Objetos ConnectionSpec
- Objetos InteractionSpec
- Objetos Record

ConnectionSpec

Un objeto ConnectionSpec se puede utilizar para especificar atributos de seguridad (como ID de usuario y contraseña) en la interacción con un servidor.

Nota: CICS ignora cualquier valor de seguridad especificado en un objeto ConnectionSpec, porque ya ha establecido un contexto de seguridad adecuado para el conector.

La clase ConnectionSpec del CCI Connector for CICS TS se llama ECIConnectionSpec.

InteractionSpec

Un objeto InteractionSpec mantiene atributos esenciales necesarios para una interacción con un servidor; por ejemplo, el nombre del programa destino. Se pasa como un argumento necesario en una llamada al método **Interaction.execute()** cuando se va a realizar una interacción concreta.

La clase InteractionSpec del CCI Connector for CICS TS se llama ECIInteractionSpec.

Record

Los objetos Record son beans que mantienen los datos intercambiados con el programa destino; pueden considerarse el equivalente de las áreas de comunicación (COMMAREA) de CICS. Los datos son accesibles mediante interfaces definidas por Record.

La Figura 26 muestra las clases de infrasetructura de CCI y las clases de entrada/salida usadas a la vez para conectar con un EIS, pasar parámetros de entrada/salida específicos de EIS y ejecutar un mandato.

```
ConnectionFactory cf = <Lookup from JNDI namespace>
ECIConnectionSpec cs = new ECIConnectionSpec();
cs.setXXX();     //Defina cualquier propiedad específica de la conexión

Connection conn = cf.getConnection(cs);
Interaction int = conn.createInteraction();
ECIInteractionSpec is = new ECIInteractionSpec();
is.setXXX();     //Defina cualquier propiedad específica de la interacción

RecordImpl in = new RecordImpl();
RecordImpl out = new RecordImpl();
int.execute(is,in,out);
int.close();
conn.close();
```

Figura 26. Interacción completa de CCI con un EIS

El CCI Connector for CICS TS:

La CICS Transaction Gateway incluye un adaptador de recursos de interfaz de llamada externa (ECI) para CICS.

El **adaptador de recursos ECI** proporciona interfaces CCI estándar que permiten a los componentes de J2EE llamar a los programas de servidor de CICS utilizando áreas de datos (COMMAREAs) para pasar información al servidor y desde este. Normalmente, estos componentes de J2EE son servlets o enterprise beans; en todos los casos, se ejecutan fuera de CICS.

CICS TS incluye el CCI Connector for CICS TS, que también brinda interfaces CCI estándar que permiten a los programas y componentes Java (por ejemplo, enterprise beans) que se ejecuten *en CICS* llamar a programas de servidor de CICS.

Un programa Java o un enterprise bean que se ejecute en CICS TS puede utilizar las clases de CCI Connector for CICS para enlazar con un programa de servidor de CICS adecuado. El programa de servidor de CICS:

- Se puede escribir en cualquiera de los lenguajes soportados por CICS.
- Debe utilizar un área de comunicaciones (COMMAREA) adecuada.
- No debe realizar ninguna entrada/salida de terminal.
- Generalmente, se ejecuta en una región CICS Transaction Server para z/OS de fondo distinta, pero, opcionalmente, puede estar en la misma región CICS que el programa o el bean Java.

El conector utiliza una llamada a `Program.link()` de JCICS para acceder al programa de servidor de fondo. Se soportan las llamadas a enlace y a enlace de programa distribuido (DPL). Este escenario se muestra en la Figura 27 en la página 341. En este ejemplo, una aplicación de cliente o un servlet Java utiliza RMI-IIOP para crear una instancia de un enterprise bean en un servidor EJB de CICS. El enterprise bean utiliza el CCI Connector for CICS TS para enlazar con un programa de servidor en una región CICS Transaction Server para z/OS de fondo.

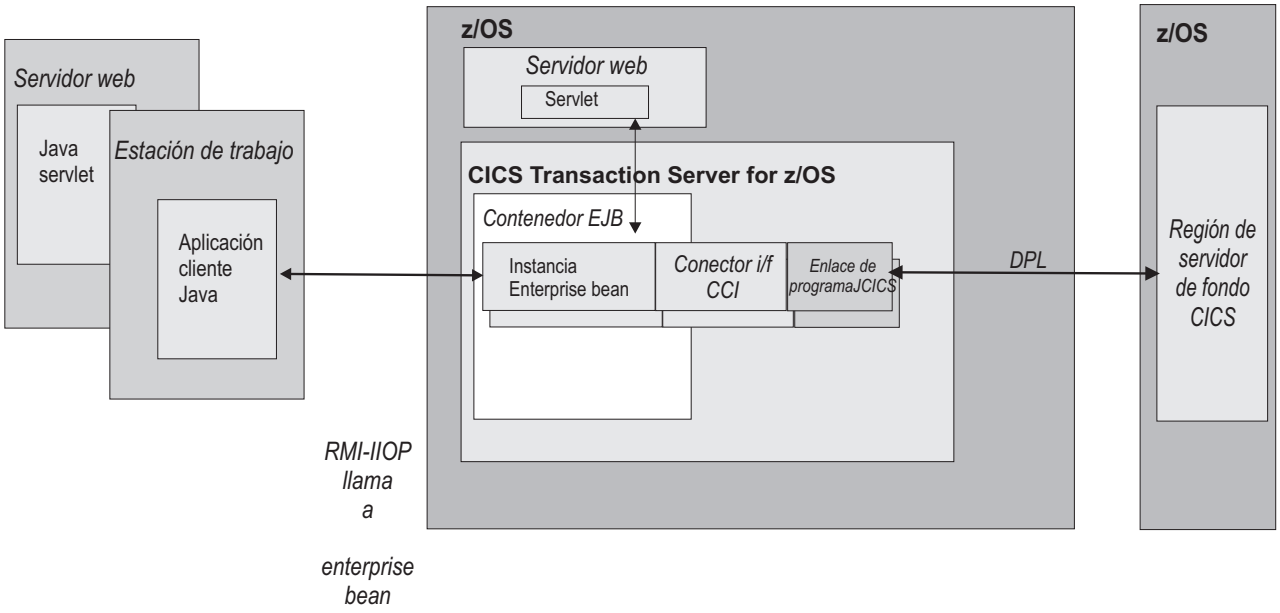


Figura 27. Un enterprise bean de CICS utiliza el CCI Connector for CICS TS para conectar con un programa de servidor de CICS.

Una aplicación de cliente o un servlet Java utiliza RMI-IIOP para crear una instancia de un enterprise bean que existe en un contenedor EJB de CICS. El enterprise bean utiliza el CCI Connector for CICS TS para enlazar con un programa de servidor en una región CICS TS para z/OS de fondo.

Para crear un enterprise bean que utilice el CCI Connector for CICS TS, el programador de Java necesita tener un conocimiento razonable de CICS (aunque algo menos que si estuviera utilizando JCICS). Sin embargo, los programadores de Java que tengan menos conocimiento de CICS pueden utilizar los enterprise beans que se creen.

El CCI Connector for CICS TS está altamente optimizado para su ejecución en CICS; hay muy poca sobrecarga si se utiliza en lugar de una llamada a `Program.link()` de JCICS.

Beneficios del CCI Connector for CICS TS:

Hay varios beneficios derivados del uso del CCI Connector for CICS TS para crear potentes componentes de servidor que aprovechen los programas CICS existentes.

1. Los enterprise beans de CICS que utilizan el conector:
   - Permiten a los programadores de aplicaciones de cliente Java, que generalmente tienen poco o ningún conocimiento de CICS, añadir la capacidad de CICS a sus aplicaciones.
   - Pueden llamarlos las aplicaciones de cliente Java y los servlets que se ejecutan en muchas plataformas. El código de cliente utilizado para llamar al bean (y mediante él, al programa de servidor de CICS) es idéntico en todas las plataformas Java. Así, por ejemplo, el cliente podría ser un enterprise bean que se ejecute en WebSphere, un servlet que se ejecute en un servidor web o una aplicación autónoma de una estación de trabajo.
   - Si está escrito correctamente, puede ser portátil, con poca o ninguna modificación, entre todos los servidores EJB que soportan la Common Client Interface (interfaz de cliente común).

2. Como la interfaz de cliente común (CCI) es un estándar sin derechos reservados, el código de la CCI que llama al programa de servidor debería ser portátil, con poca o ninguna modificación, entre la mayoría de las plataformas preparadas para Java.
3. Como el CCI Connector for CICS TS se ejecuta *dentro* de CICS, no es necesario ningún flujo de red entre el conector y CICS. Así, el rendimiento del conector es mejor que el de los conectores CCI que utilizan el adaptador de recursos ECI para acceder a programas de CICS desde fuera de CICS.
4. El uso del conector desde un bean de sesión de CICS resulta en un modelo de despliegue simple de dos niveles: Cliente → CICS TS.
5. Los programas escritos para utilizar el adaptador de recursos ECI se pueden adaptar fácilmente para utilizar el CCI Connector for CICS TS. Así, los programas cliente que antes accedían a programas de servidor de CICS desde fuera de CICS se pueden migrar para que se ejecuten dentro de CICS.

   Nota: Si se traslada un programa escrito para utilizar el adaptador de recursos ECI a utilizar el CCI Connector for CICS TS, debe volver a compilar el programa para que utilice las clases proporcionadas por CICS TS en el archivo JAR `dfjcci.jar`, en lugar de las clases de la CICS Transaction Gateway.
6. El CCI Connector for CICS TS soporta el mecanismo de política de seguridad de Java 2.

Aplicaciones de ejemplo:

CICS proporciona dos aplicaciones de ejemplo que ilustran cómo un programa Java de CICS o un enterprise bean puede utilizar el CCI Connector for CICS TS para llamar a un programa de servidor de CICS.

1. El programa de ejemplo CCI Connector. Se trata de una aplicación relativamente sencilla que muestra cómo codificar las API de CCI directamente.

   El programa de ejemplo CCI Connector ilustra cómo:
   a. Buscar una fábrica de conexiones publicada anteriormente en un espacio de nombres JNDI.
   b. Utilizar el CCI Connector for CICS TS para llamar a un programa de servidor de CICS.

   El programa de ejemplo CCI Connector se describe en "La aplicación de ejemplo CCI Connector" en la página 350.
2. La aplicación de ejemplo EJB Bank Account. Se trata de una aplicación de ejemplo más compleja que enseña cómo se pueden utilizar enterprise beans y DB2 para hacer que la información controlada por CICS esté disponible para los usuarios web. La aplicación de ejemplo implementa un enterprise bean de CICS que utiliza el CCI Connector for CICS TS para enlazar con programas COBOL de CICS de fondo. El programa COBOL extrae información de tablas de datos de DB2.

   La aplicación de ejemplo EJB Bank Account se describe en "La aplicación de ejemplo EJB Bank Account" en la página 290.

CICS también proporciona dos programas de utilidad de ejemplo que enseñan cómo:

1. Publicar una fábrica de conexiones en un espacio de nombres de JNDI (el programa de ejemplo `CICSConnectionFactoryPublish`). Esto se describe en "Publicación de una fábrica de conexiones utilizando CICSConnectionFactoryPublish" en la página 348.

2. Retirar una fábrica de conexiones publicada anteriormente desde el espacio de nombres de JNDI (el programa de ejemplo `CICSConnectionFactoryRetract`). Esto se describe en "Retirada de una fábrica de conexiones utilizando CICSConnectionFactoryRetract" en la página 350.

Utilización del CCI Connector for CICS TS

Los componentes Java de CICS que utilizan el CCI Connector for CICS TS se pueden programar de dos formas.

Acerca de esta tarea

1. Programe directamente en la implementación del conector de la interfaz de cliente común (CCI). Este método produce el mejor rendimiento.
2. Use una herramienta de desarrollo rápido de aplicaciones (RAD) que proporcione interfaces visuales y construcciones de nivel alto para programar la interfaz de cliente común del conector.

Sea cual sea el método que elija, tiene que comprender cómo utilizar el CCI Connector for CICS TS desde un componente Java que se ejecute en CICS TS.

La lógica que un enterprise bean de CICS debe utilizar para enlazar con un programa CICS de fondo se muestra en Figura 26 en la página 339. Es decir:

1. Utilice el programa de ejemplo `CICSConnectionFactoryPublish` para publicar un objeto ConnectionFactory adecuado para su uso con el CCI Connector for CICS TS en el espacio de nombres de JNDI utilizado por la región CICS local. (Consulte "Uso de los programas de utilidad de ejemplo para gestionar y adquirir una fábrica de conexiones" en la página 347).
2. Declare un objeto ConnectionFactory y defínalo para la fábrica de conexiones de CICS mediante una búsqueda JNDI.
3. Cree un objeto ECIConnectionSpec. Defina sus propiedades como necesarias (necessary).

   Nota: Este paso se incluye para fines de integridad. Sin embargo, CICS ignora cualquier ID de usuario o contraseña que se especifique en el objeto ECIConnectionSpec.
4. Utilice ConnectionFactory para crear un objeto Connection. Este objeto representa una única conexión con CICS.
5. Cree un objeto Interaction desde el objeto Connection.
6. Cree un objeto ECIInteractionSpec. Defina sus propiedades, incluido el nombre del programa destino y la modalidad —síncrona o asíncrona (synchronous y asynchornous respectivamente)— de la interacción. (En el caso de CICS TS, solo está soportada la modalidad síncrona).
7. Cree dos objetos Record para representar las áreas de comunicaciones de entrada y de salida del programa destino.
8. Ejecute el método execute del objeto Interaction, pasando ECIInteractionSpec, y los objetos Record de entrada y salida como argumentos.
9. Recupere los datos devueltos por el programa destino desde el objeto Record de salida.
10. Ejecute el método close del objeto Interaction.
11. Ejecute el método close del objeto Connection.

Nota: Para especificar la región de servidor CICS propietaria del programa con el que enlazar, utilice el recurso PROGRAM del programa de servidor. Especifique la ubicación del programa de servidor (local o remota) y, si es remota, si debe producirse direccionamiento dinámico.

Importante: Utilice el Javadoc para la API de arquitectura de CCI Connector para ayudarle a codificar las aplicaciones de CCI. También proporciona información como las excepciones utilizadas por las implementaciones de CCI. El Javadoc para las clases ECIConnectionSpec y ECIInteractionSpec específicas de CICS está en *CCI Connector for CICS TS: referencia de clases*, en el Information Center de CICS.

¿Qué clases se deben utilizar?:

¿Qué clases se deben utilizar, las clases CCI estándar del paquete `javax.resource.cci` o las clases específicas de CICS que proporciona el CCI Connector for CICS TS en el paquete `com.ibm.connector2.cics`?

Clases de infraestructura:

El CCI Connector for CICS TS proporciona implementaciones de las clases de infraestructura llamadas **ECIConnectionFactory**, **ECIConnection** y **ECIInteraction**.

Sin embargo, se deberían utilizar las clases estándar **ConnectionFactory**, **Connection** e **Interaction**, en lugar de las implementaciones específicas de CICS. Para obtener información orientativa sobre la programación de estas clases, consulte la *CICS Transaction Gateway: Programming Guide*. Para obtener información de referencia, consulte el Sun Javadoc generado desde el código fuente de las clases **ConnectionFactory**, **Connection** e **Interaction**.

Tenga en cuenta que no toda la información de la *CICS Transaction Gateway: Programming Guide* es aplicable al CCI Connector for CICS TS. Las siguientes propiedades de la clase **ConnectionFactory** (y de la clase **ECIManagedConnectionFactory** proporcionada por CICS) las ignora CICS TS:

- clientSecurity
- connectionURL (en CICS TS, siempre es `local:`)
- password
- portNumber
- serverName
- serverSecurity
- userName

Especificar un valor para cualquiera de las propiedades anteriores no causa ningún efecto.

Clases de entrada/salida:

El CCI Connector for CICS TS proporciona implementaciones de las clases de entrada/salida. Utilice estas clases específicas de CICS (ECIConnectionSpec y ECIInteractionSpec} en lugar de las clases estándar ConnectionSpec y InteractionSpec.

Para obtener información orientativa sobre la programación de clases específicas de CICS, consulte la *CICS Transaction Gateway: Programming Guide*. Para obtener información de referencia, consulte el Javadoc de CICS generado a partir de las

clases ECIConnectionSpec y ECIInteractionSpec en *CCI Connector for CICS TS: Class Reference*. A continuación se indican algunas consideraciones especiales que se aplican al CCI Connector for CICS TS.

Nota: Especificar una propiedad o un valor descrito como "no soportado por CICS TS" produce una excepción. Especificar una propiedad o un valor descrito como "ignorado por CICS TS" no tiene efecto alguno.

ECIConnectionSpec
Esta clase permite que el componente J2EE pase credenciales de seguridad distintas a las definidas por la fábrica de conexiones. Las propiedades incluyen:

Password
La contraseña para el ID de usuario especificado en UserName. Ignorado por CICS TS.

UserName
El ID de usuario que utilizar para acceder a CICS. Ignorado por CICS TS.

ECIInteractionSpec
Esta clase mantiene todos los atributos relevantes para la interacción (por ejemplo, el nombre del programa destino y la modalidad de la interacción, síncrona o asíncrona) necesarios para una interacción con CICS. Es un parámetro necesario en cada llamada a método Interaction.execute(). Sus propiedades son:

InteractionVerb
La modalidad de la llamada a CICS: síncrona o asíncrona. El CCI Connector for CICS TS soporta solo las siguientes:

SYNC_SEND_RECEIVE
Una llamada síncrona. Se utiliza para enlazar con un programa CICS.

FunctionName
El nombre del programa que ejecutar en CICS. El CCI Connector for CICS TS requiere que se especifique FunctionName.

Nota: FunctionName puede hacer referencia a un programa local o remoto. La definición PROGRAM de la región local especifica la ubicación del programa de servidor (local o remoto) y, si es remoto, si debe producirse direccionamiento dinámico o no.

ExecuteTimeout
El valor de tiempo de espera para interacciones con CICS.

0 Sin tiempo de espera. Se trata del valor predeterminado y es el único valor que soporta CICS TS.

Un entero positivo
La cantidad de tiempo en milisegundos. Ignorado por CICS TS.

CommareaLength
La duración del área de comunicaciones (COMMAREA) que se pasa a CICS dentro del registro de entrada. Si no se proporciona, el valor predeterminado utilizado por el CCI Connector for CICS TS es la duración de los datos del registro de entrada.

ReplyLength
La cantidad de datos que se quiere recuperar de CICS. Si su enterprise bean o su componente Java solo necesita una pequeña cantidad de una gran COMMAREA devuelta, puede utilizar este valor para recortar el

ancho de banda de la red. Si no se proporciona, el valor predeterminado es recibir todos los datos en la COMMAREA.

Nota: Se recomienda no definir ReplyLength. Como el CCI Connector for CICS TS siempre se ejecuta en modalidad local —es decir, el enterprise bean o el componente Java que llama al conector se ejecuta en la misma región CICS que el propio conector— no hay flujo de red que considerar y, por lo tanto, no es necesario recibir menos que la respuesta completa.

Record
Para entrada o salida, el CCI Connector for CICS TS solo soporta clases Record que implementen la interfaz javax.resource.cci.Streamable. Esto permite al conector leer y grabar las secuencias de bytes que conforman las COMMAREA de CICS directamente en los objetos Record proporcionados en el método **execute()** de ECIInteraction y desde dichos objetos.

Para obtener más información sobre la utilización de la interfaz javax.resource.cci.Streamable para compilar registros de entrada y recuperar matrices de desde registros de entrada, consulte la *CICS Transaction Gateway: Programming Guide*.

Conversión de datos del CCI Connector for CICS TS

Para representar datos de texto, los programas Java siempre utilizan el juego de caracteres Unicode, mientras que los programas CICS TS utilizan EBCDIC.

Cuando un programa Java o un enterprise bean llama a un programa de servidor de CICS TS, cualquier valor de texto en el área de comunicaciones del programa de servidor debe convertirse de Unicode a EBCDIC en la entrada y de EBCDIC a Unicode en la salida. Sin embargo, el CCI Connector for CICS TS gestiona esta conversión de datos automáticamente. Al convertir a Unicode y desde este, la llamada a Program.link() de JCICS emitida por el conector utiliza, como sistema de codificación alternativo, el sistema de codificación del entorno de ejecución; como el conector se ejecuta en z/OS, el sistema de codificación alternativo es EBCDIC.

Nota: De forma predeterminada, los objetos Record que se pasan al método Interaction.execute() del conector utilizan la página de códigos de EBCDIC utilizada por el entorno de ejecución del conector.

Instalación del CCI Connector for CICS TS

Compilación de aplicaciones CCI

Para compilar una aplicación que utilice el CCI Connector for CICS TS, debe incluir estos archivos JAR que proporciona CICS en su vía de acceso de clases Java:

connector.jar
Las API de CCI, que necesitan todas las aplicaciones de CCI.

dfjcci.jar
Las implementaciones de CICS TS de las API de CCI.

Al instalar CICS, `connector.jar` se instala en el directorio `%JAVA_HOME%/standard/jca` z/OS UNIX (donde %JAVA_HOME% es el valor del parámetro JAVADIR en el trabajo de instalación DFHISTAR de CICS); `dfjcci.jar` se instala en el directorio `/usr/lpp/cicsts/cicsts42/lib` (donde `cicsts42` es el valor del parámetro USSDIR en el trabajo de instalación DFHISTAR).

Uso de los programas de utilidad de ejemplo para gestionar y adquirir una fábrica de conexiones

Acerca de esta tarea

CICS proporciona tres programas de ejemplo que ilustran cómo:

1. Publicar una fábrica de conexiones en un espacio de nombres de JNDI (el programa de ejemplo `CICSConnectionFactoryPublish`). Puede utilizar el programa de ejemplo para crear un objeto **ConnectionFactory** adecuado para su uso con el CCI Connector for CICS TS, y para publicarlo en el espacio de nombres de JNDI utilizado por la región CICS local. Un enterprise bean o un programa Java, que se ejecute en CICS, luego puede realizar una búsqueda JNDI para obtener una referencia a la fábrica de conexiones.

   Este ejemplo se describe en "Publicación de una fábrica de conexiones utilizando CICSConnectionFactoryPublish" en la página 348.
2. Retirar una fábrica de conexiones publicada anteriormente desde el espacio de nombres de JNDI (el programa de ejemplo `CICSConnectionFactoryRetract`). Este ejemplo se describe en "Retirada de una fábrica de conexiones utilizando CICSConnectionFactoryRetract" en la página 350.
3. Buscar una fábrica de conexiones en el espacio de nombres de JNDI (el programa de ejemplo CCI Connector). Este programa de ejemplo también muestra cómo utilizar el CCI Connector for CICS TS para llamar a un programa de servidor de CICS. Esto se describe en "La aplicación de ejemplo CCI Connector" en la página 350.

Mediante los programas de ejemplo `CICSConnectionFactoryPublish` y `CICSConnectionFactoryRetract`, puede crear, publicar y gestionar una fábrica de conexiones independientemente de las aplicaciones que la utilicen.

Para utilizar los programas de ejemplo, necesita un servidor de nombres configurado adecuadamente. Si tiene que configurar un servidor de nombres, consulte "Habilitar referencias de JNDI" en la página 396 y "Especificación de la ubicación del servidor de nombres JNDI" en la página 396.

Instalación de los programas de ejemplo de publicación y retirada:

Esta sección describe cómo instalar los programas `CICSConnectionFactoryPublish` y `CICSConnectionFactoryRetract`.

Acerca de esta tarea

Cómo instalar la aplicación CCI Connector se describe en "Instalación del programa de ejemplo CCI Connector" en la página 352.

El archivo JAR proporcionado por CICS `CICSCCISamples.jar` contiene los archivos de objeto (.class) para los programas de ejemplo. CICS instala `CICSCCISamples.jar` en el directorio `/usr/lpp/cicsts/cicsts42/samples/cci` (donde `/usr/lpp/cicsts/cicsts42` es el directorio de instalación para los archivos de CICS en z/OSUNIX). También se instalan en el directorio `/usr/lpp/cicsts/cicsts42/samples/cci` los archivos de origen (.java) de los programas.

Para instalar los programas `CICSConnectionFactoryPublish` y `CICSConnectionFactoryRetract`:

Procedimiento

1. Añada el archivo JAR que contiene los programas, `/usr/lpp/cicsts/cicsts42/samples/cci/CICSCCISamples.jar`, a la sentencia CLASSPATH_SUFFIX en el perfil de JVM que utilizarán los programas. Tal y como se entregan, los programas de ejemplo utilizan el perfil de JVM de ejemplo proporcionado por CICS, DFHJVMPR, que es el predeterminado si no se especifica ningún perfil de JVM en la definición de recurso del programa. CICS instala DFHJVMPR en el directorio `/usr/lpp/cicsts/cicsts42/JVMProfiles`.
2. Coloque su versión editada de DFHJVMPR en el directorio de z/OS UNIX especificado en el parámetro de inicialización del sistema **JVMPROFILEDIR**. (En una instalación de CICS predeterminada, **JVMPROFILEDIR** especifica `/usr/lpp/cicsts/cicsts42/JVMProfiles`).
3. Utilice CEDA para instalar transacciones CCPB y CCRT desde el grupo DFH$CCI.
4. Utilice CEDA para instalar los programas DFJ$CCPB y DFJ$CCRT desde el grupo DFH$CCI.

   Nota: Si su región CICS utiliza el programa de instalación automática (autoinstall), este último paso no es necesario.

Resultados

Publicación de una fábrica de conexiones utilizando CICSConnectionFactoryPublish:

El programa `CICSConnectionFactoryPublish` realiza las siguientes tareas.

1. Obtiene el contexto JNDI inicial de la región CICS.
2. Comprueba si existe un subcontexto `ConnectionFactory` en la estructura de contexto.
3. Si el subcontexto de `ConnectionFactory` no existe, lo crea.
4. Si la fábrica de conexiones `ConnectionFactory/CICSConnectionFactory` aún no se ha publicado (enlazado) en el servidor de nombres, la publica.

El nombre predeterminado de la fábrica de conexiones, según lo define la versión provista del programa `CICSConnectionFactoryPublish`, es `CICSConnectionFactory`. El nombre predeterminado del subcontexto JNDI en el que se publica la fábrica de conexiones es `ConnectionFactory`. Si se edita el código fuente del programa `CICSConnectionFactoryPublish`, es posible cambiar:

- El nombre de la fábrica de conexiones.
- El subcontexto JNDI.
- El nombre de la transacción de duplicación en la que el programa se ejecuta en la región remota, si el programa de servidor al que está vinculada es remoto. Sin embargo, el modo recomendado de especificar el programa de duplicación es en la definición local de PROGRAM del programa de servidor.

Para obtener instrucciones sobre cómo realizar los cambios, consulte los comentarios en el código fuente.

Si cambia el nombre de la fábrica de conexiones o el del subcontexto, recuerde realizar el mismo cambio en los tres programas de ejemplo.

Ejecución del programa:

Para publicar (enlazar) una fábrica de conexiones adecuada para utilizar con el CCI Connector for CICS TS en el servidor de nombres JNDI de CICS, ejecute la transacción CCPB.

A no ser que haya modificado el programa `CICSConnectionFactoryPublish`, la fábrica de conexiones tendrá el nombre `CICSConnectionFactory` y se publicará en el subcontexto `ConnectionFactory` del espacio de nombres del servidor de JNDI.

El siguiente mensaje aparece en la pantalla:

```
ccpb - ConnectionFactory published to JNDI successfully.
```

Nota: Si ya se ha publicado una fábrica de conexiones con el mismo nombre y el mismo subcontexto para el servidor JNDI (y no se ha retirado), aparece un mensaje distinto:

```
ccpb - The ConnectionFactory is already published to JNDI.
```

Si asumimos que la fábrica de conexiones se publica correctamente, se envía la siguiente salida a **stdout**:

```
************************************************************************************************
**** CICSConnectionFactoryPublish: Started
**** CICSConnectionFactoryPublish: Binding ConnectionFactory ConnectionFactory/CICSConnectionFactory
**** CICSConnectionFactoryPublish: ConnectionFactory bound to JNDI
**** CICSConnectionFactoryPublish: Ended
************************************************************************************************
```

Figura 28. Salida stdout de la transacción CCPB para publicar una fábrica de conexiones con nombre y subcontexto predeterminados

No se recomienda ejecutar `CICSConnectionFactoryPublish` como un programa de PLTPI ni enlazar con él desde un programa de PLTPI. Esto se debe a que si no hay ninguna JVM disponible, el tiempo de inicio de CICS se alargará.

Búsqueda de una fábrica de conexiones:

Este código de ejemplo muestra cómo buscar una fábrica de conexiones publicada anteriormente en un espacio de nombres JNDI utilizado por CICS.

```
// Declare un objeto ConnectionFactory.
ConnectionFactory cf = null;

try{
    // Pruebe el contexto JNDI inicial.
    javax.naming.Context ic = new javax.naming.InitialContext();

    // Realice la búsqueda, difundiendo la CICSConnectionFactory devuelta para el tipo
    // ConnectionFactory.
    cf = (ConnectionFactory)ic.lookup("ConnectionFactory/CICSConnectionFactory");

    // Utilice la fábrica de conexiones para crear una conexión con CICS.
    Connection eciConn = (Connection)cf.getConnection();
}
catch (Exception e){
    // Se produjo un error en la búsqueda o se especificó una fábrica de conexiones que
       no se ha publicado.
    // Proceso de la excepción.
}
```

Esto se ilustra en la aplicación CCI Connector; consulte "La aplicación de ejemplo CCI Connector" en la página 350.

Retirada de una fábrica de conexiones utilizando CICSConnectionFactoryRetract:

Para retirar (desenlazar) una fábrica de conexiones que haya publicado, ejecute la transacción CCRT. A no ser que haya modificado el programa `CICSConnectionFactoryRetract`, la fábrica de conexiones que hay que retirar será `CICSConnectionFactory`, en el subcontexto `ConnectionFactory` del espacio de nombres del servidor de JNDI.

El siguiente mensaje aparece en la pantalla:

```
ccrt - ConnectionFactory retracted from JNDI successfully.
```

Nota: Si la fábrica de conexiones indicada en el programa `CICSConnectionFactoryRetract` no existe en el servidor de JNDI (por ejemplo, puede que ya se haya retirado), aparece un mensaje distinto:

```
ccrt - unable to locate ConnectionFactory on JNDI.
```

Si asumimos que la fábrica de conexiones se retira de forma normal, se envía la siguiente salida a **stdout**:

```
********************************************************************************
**** CICSConnectionFactoryRetract: Started
**** CICSConnectionFactoryRetract: Unbinding ConnectionFactory/CICSConnectionFactory
**** CICSConnectionFactoryRetract: ConnectionFactory/CICSConnectionFactory unbound
**** CICSConnectionFactoryRetract: Ended
********************************************************************************
```

Figura 29. Salida de Stdout de la transacción CCRT para retirar una fábrica de conexiones con nombre y subcontexto predeterminados

No se recomienda ejecutar `CICSConnectionFactoryRetract` como un programa PLTSD ni enlazar con él desde un programa PLTSD. Esto se debe a que el tiempo necesario para la conclusión de CICS se alargará.

La aplicación de ejemplo CCI Connector

La aplicación de ejemplo CCI Connector es una aplicación relativamente sencilla que muestra cómo codificar las API de CCI directamente.

Ilustra cómo se hace lo siguiente:

1. Buscar una fábrica de conexiones publicada anteriormente en un espacio de nombres JNDI.
2. Utilizar el CCI Connector for CICS TS para llamar a un programa de servidor de CICS.

La aplicación de ejemplo consta de:

- Un programa Java de CICS.
- Un registro personalizado que enseña a utilizar la interfaz javax.resource.cci.Streamable.
- Un programa de servidor COBOL de CICS.

La aplicación de ejemplo funciona de esta manera:

1. Un usuario lanza la aplicación ejecutando la transacción CCCI desde un terminal de CICS.
2. El programa Java de CICS, `CICSCCISample` (DFJ$CCIC), se inicia. El programa Java:
   a. Pide al usuario que especifique una secuencia de números decimales aleatorios sin ordenar.

b. Realiza una búsqueda JNDI del servidor de nombres para obtener una fábrica de conexiones de CICS.
c. Si no se ha publicado una fábrica de conexiones de CICS para el servidor de nombres, crea una mediante programación.
d. Utiliza la fábrica de conexiones para crear una conexión con CICS.
e. Crea un objeto Interaction (interacción) desde el objeto **Connection** (conexión) y define las propiedades de la interacción (incluido el nombre del programa destino) mediante un objeto ECIInteractionSpec.
f. Utiliza el método Interaction.execute para enlazar con el programa COBOL, DFH$0CCIS, y pasa como entrada (en un objeto Record personalizado) la secuencia del usuario de números sin ordenar, además del objeto ECIInteractionSpec.

3. El programa COBOL ordena los números en orden ascendente y devuelve la secuencia ordenada en su COMMAREA de salida.
4. El programa Java recupera la salida del programa COBOL desde el objeto **Record** (registro) de salida y visualiza la lista ordenada en el terminal del usuario.

La Figura 30 en la página 352 muestra los componentes de la aplicación de ejemplo.

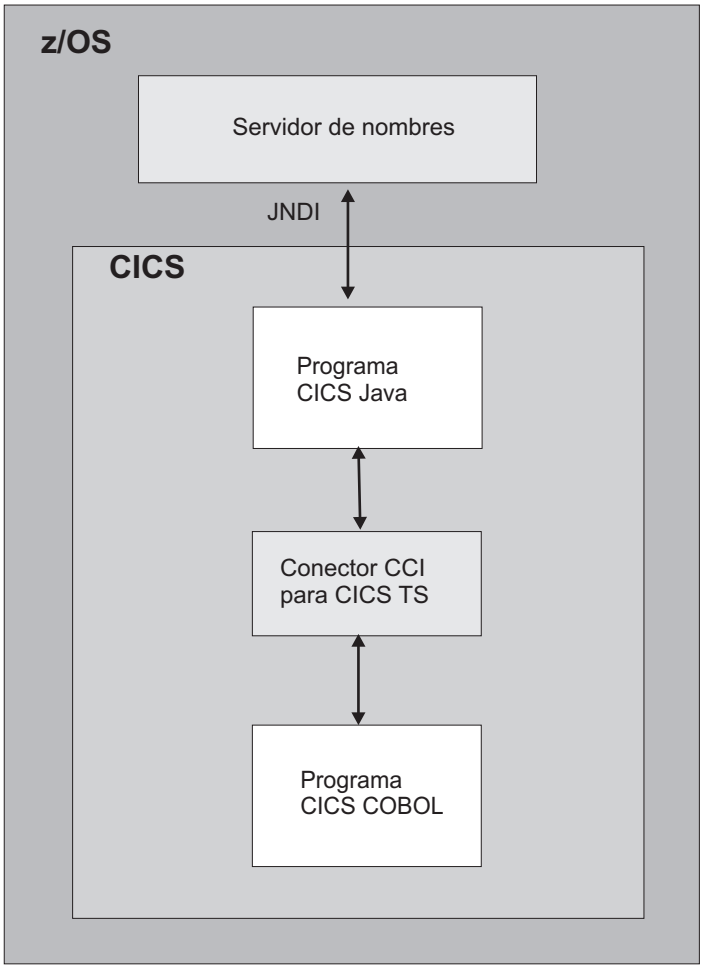


Figura 30. Visión general de la aplicación de ejemplo CCI Connector. Los elementos principales de la aplicación de ejemplo son un programa Java de CICS y un programa de servidor COBOL de CICS. El programa Java utiliza el CCI Connector for CICS TS para enlazar con el programa de servidor COBOL. La fábrica de conexiones CICS puede publicarse en un COS Naming Server o en un servidor de nombres LDAP.

Requisitos para el programa de ejemplo CCI Connector:

Para permitir que el programa de ejemplo CCI Connector obtenga una fábrica de conexiones de CICS realizando una búsqueda JNDI, necesita un servidor de nombres que soporte la Java Naming and Directory Interface (JNDI) versión 1.2 o posterior.

El modo de configurar uno se describe en "Acciones necesarias en z/OS o Windows NT" en la página 259. Puede utilizar un COS Naming Server o un servidor LDAP.

Sin embargo, si el programa de ejemplo no se puede conectar con el servidor de nombres, o si no se ha publicado una fábrica de conexiones de CICS para dicho servidor de nombres, el programa de ejemplo crea la fábrica de conexiones mediante programación. Por lo tanto, para ser estrictos, un servidor de nombres no es un requisito para ejecutar el programa de ejemplo.

Instalación del programa de ejemplo CCI Connector:

Acerca de esta tarea

Procedimiento

1. Si aún no lo ha hecho al ejecutar los ejemplos `CICSConnectionFactoryPublish` y `CICSConnectionFactoryRetract`, localice el archivo JAR que contiene los programas de ejemplo, `/usr/lpp/cicsts/cicsts42/samples/cci/CICSCCISamples.jar`, donde `/usr/lpp/cicsts/cicsts42` es el directorio de instalación para los archivos de CICS en z/OS UNIX. Añada este archivo JAR a la sentencia CLASSPATH_SUFFIX en el perfil de JVM que utilizarán los programas. Tal y como se suministran, los programas de ejemplo utilizan el perfil de JVM de ejemplo proporcionado por CICS, DFHJVMPR, que es el predeterminado si no se especifica ningún perfil de JVM en la definición de recurso del programa.

   CICS instala DFHJVMPR en el directorio `/usr/lpp/cicsts/cicsts42/JVMProfiles`.

   Coloque su versión editada de DFHJVMPR en el directorio de z/OS UNIX especificado en el parámetro de inicialización del sistema **JVMPROFILEDIR**.
2. Asegúrese de que los archivos `connector.jar` y `dfjcci.jar` están en la vía de acceso de clase estándar.

   Nota: Cuando se instala CICS, `connector.jar` se instala en el directorio `%JAVA_HOME%/standard/jca` y `dfjcci.jar` se instala en el directorio `/usr/lpp/cicsts/cicsts42/lib`, tal y como se describe en "Compilación de aplicaciones CCI" en la página 346. El directorio `/usr/lpp/cicsts/cicsts42/lib` está en la vía de acceso de clases base creada por CICS, que no es visible en los perfiles de JVM, por lo que este directorio siempre se incluye.
3. Asegúrese de que el servidor de nombres se está ejecutando.
4. Utilice el programa `CICSConnectionFactoryPublish` para crear un objeto ConnectionFactory para su uso con el CCI Connector for CICS TS y para publicarlo en el espacio de nombres. Consulte "Publicación de una fábrica de conexiones utilizando CICSConnectionFactoryPublish" en la página 348.
5. Utilice CEDA para instalar la transacción CCCI desde el grupo DFH$CCI.
6. Utilice CEDA para instalar definiciones de los programas Java y COBOL de CICS. Instale los programas DFJ$CCIC y DFH0CCIS desde el grupo DFH$CCI.

   Nota: Si su región CICS utiliza el programa de instalación automática (autoinstall), este paso no es necesario.

Prueba del programa de ejemplo:
Acerca de esta tarea

Para probar el programa de ejemplo CCI Connector:

1. Lance la transacción CCCI en un terminal CICS.
2. El programa de ejemplo le pide que especifique algunos números. Escriba al menos cinco números decimales, separados por espacios, y pulse la tecla Intro. (Cada número debe tener cinco dígitos o menos, y no deben estar ordenados por tamaño).
3. El programa de ejemplo escribe la lista ordenada de números en la pantalla y la graba en **stdout**. Por ejemplo, si ha escrito los números 54, 3, 77, 55 y 19, la pantalla tendría un aspecto similar a este:

```
CCCI - CCI sample transaction starting.

A Connection object has been instantiated.
```

```
An Interaction object has been instantiated.

Enter a series of numbers:  54 3 77 55 19

An InteractionSpec object has been instantiated.

Connecting to program DFH0CCIS by invoking execute() on Interaction object.

Commarea sent:    54    3   77   55   19*

Commarea returned:     3   19   54   55   77*

CCCI - CCI sample transaction finished.
```

Determinación de problemas

Puede utilizar los mensajes del CCI Connector for CICS TS y el rastreo de CICS para diagnosticar problemas.

Mensajes de CCI Connector for CICS TS:

Los mensajes de CICS relacionados con el CCI Connector for CICS TS se describen en el manual *Mensajes y códigos de CICS Vol 1*.

Rastreo del CCI Connector for CICS TS:

Los puntos de rastreo de CICS relacionados con el conector están en el rango EJ 0600 - EJ 06FF.

Estos se describen en Descripción general de las entradas de rastreo en Entradas de rastreo.

Para controlar la salida de información de rastreo de CICS desde el conector, utilice el control de rastreo de CICS de la manera normal.

Actualización del CICS Connector for CICS TS al CCI Connector for CICS TS

Si dispone de aplicaciones existentes que utilicen el CICS Connector for CICS TS, debe actualizarlas para que utilicen el CCI Connector for CICS TS en su lugar.

En Tabla 20 se resumen las opciones de actualización para los componentes Java de CICS que utilicen el CICS Connector for CICS TS o el CCI Connector for CICS TS, e indica una solución preferida para cada caso.

Tabla 20. Vía de acceso de actualización sugerida para componentes Java de CICS que utilicen los conectores CCF o CCI de CICS

Conector utilizado por el programa actual	**Interfaz de conector utilizada por el programa actual**	**Estado en CICS TS 4.2**	**Estrategia de actualización sugerida**
CICS Connector for CICS TS	API de CICS Transaction Gateway (**ECIRequest**)	No está soportada.	La API de CICS Transaction Gateway ya no está soportada. Realice cambios técnicos para utilizar el CCI Connector for CICS TS. Programe el conector directamente o por medio de una herramienta de desarrollo rápido de aplicaciones (RAD) que lo soporte.

Tabla 20. Vía de acceso de actualización sugerida para componentes Java de CICS que utilicen los conectores CCF o CCI de CICS (continuación)

Conector utilizado por el programa actual	Interfaz de conector utilizada por el programa actual	Estado en CICS TS 4.2	Estrategia de actualización sugerida
CICS Connector for CICS TS	CCF, programada directamente con VAJ Enterprise Access Builder o similar	No está soportada.	La CCF es sustituida por la CCI. Realice cambios técnicos para utilizar el CCI Connector for CICS TS, que funciona mejor que el CICS Connector for CICS TS y utiliza una interfaz estándar de la industria. Programe el conector directamente o por medio de una herramienta RAD que lo soporte. **Nota:** Es posible programar el CCI Connector for CICS TS utilizando el VAJ Enterprise Access Builder, pero esto no se recomienda porque VAJ/EAB ya no está soportado.
CCI Connector for CICS TS	CCI, programada directamente	Soportada.	La CCI se puede utilizar indefinidamente. Programar la CCI directamente logra el mejor rendimiento.
CCI Connector for CICS TS	CCI, programada con VAJ Enterprise Access Builder o similar	Soportada.	Para continuar utilizando VAJ/EAB, hay que realizar cambios en la aplicación.

Gestión de problemas de enterprise beans de CICS

Esta sección contiene información orientativa para la gestión de problemas al configurar y utilizar el soporte de enterprise beans de CICS.

Consulte Descripción general de determinación de problemas en Determinación de problemas para obtener orientación sobre los aspectos más generales de la determinación y el diagnóstico de problemas de CICS.

- "Problemas de configuración de enterprise beans de CICS"
- "Uso de diagnóstico en tiempo de ejecución del servidor EJB" en la página 357
- "Utilización de diagnóstico en tiempo de ejecución del cliente EJB" en la página 358
- "Problemas de versión de clase con RMI-IIOP" en la página 360
- "Utilización de mandatos de rastreo y de capacidad de servicio de EJB" en la página 361

Problemas de configuración de enterprise beans de CICS

Si tiene dificultades para configurar el servidor EJB de CICS, el problema podría estar relacionado con la configuración básica de CICS Java. Intente ejecutar la aplicación de ejemplo HelloWorld de Java. Si esto tampoco funciona, apunta a un problema en la configuración de la JVM más que en cualquier otra cosa.

Métodos que requieren varios procesadores de solicitudes:

Si una única ejecución de un método de enterprise bean necesita más de un procesador de solicitudes, su aplicación podría sufrir problemas de punto muerto.

Acerca de esta tarea

(Se puede decir que un método "necesita más de un procesador de solicitudes" si llama a uno o más métodos distintos, generalmente remotos, cada uno de los cuales debe ejecutarse en un procesador de solicitudes distinto). Los puntos

muertos pueden deberse a que todos los procesadores de solicitudes necesarios para satisfacer el método se vean forzados a esperar a una JVM cuando no se permitan más JVM. Esto se puede deber a dos razones:

1. En este caso sencillo, el número máximo de JVM permitido que exista simultáneamente en CICS (**MAXJVMTCBS**) es menor que el número de procesadores de solicitudes para atender la solicitud de método.
2. En el caso complejo:
   - CICS procesa varias solicitudes a la vez.
   - Todas las solicitudes esperan otra JVM.
   - Todas las JVM permitidas se están utilizando actualmente.

Evitar el caso sencillo es fácil, evitar el complejo es más difícil. Es necesario asegurarse siempre de que hay suficientes JVM libres para permitir satisfacer al menos la necesidad de instancias de procesador de solicitudes de un método.

El número máximo de JVM simultáneas disponible para un método de bean se define mediante el atributo MAXACTIVE de la definición de TRANCLASS para la transacción del procesador de solicitudes. El número máximo de JVM simultáneas disponible para CICS se define mediante el parámetro de inicialización del sistema **MAXJVMTCBS**.

Para eliminar la posibilidad de puntos muertos causados por métodos de bean que utilizan varios procesadores de solicitudes:

1. Siempre que sea coherente con los requisitos de su aplicación, trate de minimizar el número de procesadores de solicitudes que necesita cada método, preferiblemente a uno. Si puede reducir los requisitos de todos los métodos, en todas las aplicaciones, a un único procesador de solicitudes, no tiene que hacer más.
2. Si no es posible reducir los requisitos de todos los métodos a un único procesador de solicitudes, descubra cuál es su "peor caso": es decir, el método de bean que requiere más procesadores de solicitudes para completarlo correctamente.
3. Cree una nueva definición TRANCLASS. Esta clase de transacción se aplicará a la transacción del procesador de solicitudes en el que se ejecutarán los métodos de bean que necesitan varios procesadores de solicitudes.
4. En la definición de TRANCLASS, defina el valor de MAXACTIVE mediante la siguiente fórmula:

   ```
   MAXACTIVE <= ((MAXJVMTCBS - n) / (n - 1)) + 1
   ```

   donde `n` es el número máximo de procesadores de solicitudes necesarios para su método de "peor caso".

   Si el resultado de este cálculo es un valor decimal, redondéelo al número entero más cercano (inferior).
5. Cree nuevas definiciones TRANSACTION y REQUESTMODEL:
   a. Cree una nueva definición TRANSACTION para la transacción del procesador de solicitudes en el que se ejecutarán los métodos de bean que necesitan varios procesadores de solicitudes. (El modo más fácil para hacerlo es copiar la definición de la transacción predeterminada CIRP del procesador de solicitudes y modifcarla). En la opción TRANCLASS, especifique el nombre de la nueva clase de transacción.
   b. Cree una o más definiciones REQUESTMODEL. Entre ellas, sus nuevas definiciones REQUESTMODEL deben cubrir todas las solicitudes que pueden recibirse para métodos de bean que necesitan varios procesadores

de solicitudes. En la opción TRANSID de las definiciones REQUESTMODEL, especifique el nombre de la nueva transacción.

Uso de diagnóstico en tiempo de ejecución del servidor EJB

El servidor EJB proporciona diagnóstico en tiempo de ejecución para ayudarle a diagnosticar y resolver problemas. Esto incluye mensajes de error, rastreo de JVM y la arquitectura de depurador de plataforma Java (JPDA).

Errores y mensajes de enterprise beans de CICS:

Esta es una lista de lugares en los que buscar mensajes de error desde CICS.

Mensajes del dominio Enterprise Java (DFHEJnnnn)
CICS emite un gran número de mensaje informativos, de aviso y de error desde el dominio de Enterprise Java. La mayoría de estos se dirigen a las colas de datos transitorias CEJL y CJRM y otros se envían a la consola. Consulte el manual *Mensajes y códigos de CICS Vol 1* para obtener un listado completo.

Mensajes de JVM (DFHSJnnnn) de CICS
Estos son mensajes emitidos por la JVM de CICS. La mayoría se dirigen a la cola de datos transitoria CSMT. Consulte el manual *Mensajes y códigos de CICS Vol 1* para obtener un listado completo.

Mensajes de la herramienta de despliegue de desarrollo (DFHADnnnn) de CICS
Estos son mensajes emitidos por esta herramienta y dirigidos a CICS como mensajes de SYSPRINT. Consulte el manual *Mensajes y códigos de CICS Vol 1* para obtener un listado completo.

Códigos de terminación anómala de CICS

- De AJMA a AJM9 son emitidos por la JVM de CICS
- De AJ01 a AJ99 son emitidos por el derivador de clase de configuración del entorno Java.

Consulte el manual *Mensajes y códigos de CICS Vol 1* para obtener un listado.

Rastreo de JVM:

Las máquinas virtuales Java (JVM) tienen su propio recurso de rastreo interno. El rastreo de JVM puede ayudar en el diagnóstico de problemas de la JVM. El rastreo de JVM puede producir una gran cantidad de salidas; por lo tanto, actívelo solo para transacciones especiales, en lugar de activarlo de forma global para todas las transacciones.

"Definición y activación del rastreo para JVM en agrupación" en la página 203 explica las distintas formas de activar el rastreo de JVM y de modificar sus opciones.

Cuando se activa el rastreo de JVM, cada punto de rastreo de JVM que se genera aparece como una instancia de un punto de rastreo de CICS en el dominio SJ.

Además de las opciones de rastreo de JVM, se pueden utilizar los puntos de rastreo estándar para el dominio SJ, en los niveles de rastreo de CICS 0, 1 y 2, para el rastreo de las acciones que CICS realiza para configurar y gestionar JVM y la memoria caché de clase compartida.

Java Platform Debugger Architecture (JPDA):

La JVM en CICS soporta la Java Platform Debugger Architecture (JPDA), que es el mecanismo de depuración estándar incluido en la plataforma Java 2.

La JVM en CICS soporta la Java Platform Debugger Architecture (JPDA), que es el mecanismo de depuración estándar incluido en la plataforma Java 2. Esta arquitectura proporciona un conjunto de API que permiten adjuntar un depurador remoto a una JVM. Hay disponibles distintos depuradores de terceros que aprovechan JPDA y que se pueden utilizar para adjuntarlos a una JVM (o para depurar dicha JVM) que esté ejecutando un enterprise bean, un objeto CORBA o un programa Java de CICS. Normalmente, el depurador proporciona una interfaz gráfica de usuario que se ejecuta en una estación de trabajo y que permite seguir el flujo de la aplicación, definiendo puntos de interrupción y revisando el código fuente de la aplicación, así como examinando los valores de las variables.

Consulte “Depuración de una aplicación Java” en la página 206 para obtener pautas sobre cómo configurar y utilizar un depurador por la JVM de CICS.

Para localizar información sobre las aplicaciones compatibles con JPDA y JPDA, vaya al sitio web Oracle Technology Network Java y busque *Java Platform Debugger Architecture* para localizar la página de inicio de JPDA.

Utilización de diagnóstico en tiempo de ejecución del cliente EJB

La mayoría de los mensajes de error que emite el cliente son de uso limitado si el problema se encuentra en CICS, pero a veces puede obtenerse información útil del cliente, y este es un lugar evidente para empezar.

A continuación mostramos algunas de las excepciones más útiles del cliente:

`NoClassDefFoundException` y `ClassNotFoundException`
: Si el cliente emite alguna de estas excepciones, probablemente haya algo que falte o que esté corrompido en la vía de acceso de clases del lado de cliente. La excepción debe brindarle una buena indicación de qué clase falta, y a partir de aquí tal vez pueda averiguar qué archivo JAR añadir a la vía de acceso de clases. Recuerde que necesita `j2ee.jar` y el archivo JAR completamente desplegado en la vía de acceso de clases. Es poco probable que CICS emita más información adicional útil para estos problemas.

  NoClassDefFoundError:javax/ejb/HomeHandle
  : Esto indica que una aplicación cliente no tiene clases EJB de nivel 1.1 disponibles para la vía de acceso de clases. Asegúrese de que `j2ee.jar` está disponible.

ObjectNotFoundException
: Esta excepción puede indicar que un bean de sesión ha excedido el tiempo de espera o que se ha realizado un intento por utilizar el bean de sesión en dos o más transacciones simultáneas.

`RemoteException`
: Esto indica un problema en la aplicación de servidor y, a menudo, contiene una excepción anidada que brinda más información. Estas incluyen:

  NoClassDefFoundError
  : Esto señala que falta un archivo JAR en el lado del servidor. Compruebe la consola del sistema CICS y los archivos estándar de errores y salida de la JVM para obtener información adicional.

  CORBA.INTERNAL
  : Esto indica una anomalía en la aplicación del lado del servidor

fuera de la JVM (por ejemplo, en un programa COBOL al que ha llamado un enterprise bean). Compruebe la consola del sistema CICS para obtener más información.

Excepciones CORBA:

Estas excepciones a veces pueden proporcionar información útil.

El **estado de terminación** puede tener uno de estos valores:

- **No** significa que el servidor definitivamente no completó de ejecutar el método invocado correctamente.
- **Yes** (Sí) significa que la operación invocada en el servidor sí que se completó.
- **Maybe** (Quizá) significa que el cliente no puede determinar si la operación se completó en el servidor o no.

Si el estado de terminación es **Yes**, puede estar seguro de que el cliente encontró algo para ejecutar en un servidor (sin embargo, si el JNDI/IOR es incorrecto, puede que no haya sido el enterprise bean correcto o en la región CICS esperada). Generalmente encontrará más información útil en la salida de CICS sobre por qué la llamada a método no se realizó correctamente.

Algunas de las excepciones CORBA más comunes recibidas por el cliente son:

org.omg.CORBA.COMM_FAILURE

Esto se puede producir en una de las siguientes situaciones:

- El servidor de nombres de JNDI no se está ejecutando (si es en una búsqueda JNDI)
- El enterprise bean no se ha publicado para el servidor de nombres de JNDI.
- La región CICS no está activada.
- TCPIPSERVICE no está instalado o no está abierto (en el caso de invocaciones de método en CICS).

org.omg.CORBA.INTERNAL

Generalmente se debe a una terminación anómala o a una anomalía de la aplicación del lado del servidor. Busque en la consola de CICS para obtener más información.

org.omg.CORBA.INVALID_TRANSACTION

Esto puede deberse a problemas de interoperatividad de transacción entre un servidor de aplicaciones web y CICS.

Existen varios protocolos para el soporte de transacciones distribuidas. El entorno de CICS enterprise Java soporta solo el protocolo Servicio de transacción de objetos (OTS) de CORBA. Sin embargo, algunos servidores de aplicaciones web compatibles con J2EE (como WebSphere versión 4) no utilizan este protocolo o no lo utilizan de forma predeterminada. (Las versiones de WebSphere Application Server desde la versión 5 en adelante no se ven afectadas por este problema).

Si hay objetos en su servidor de aplicaciones web que llaman a enterprise beans de CICS dentro del ámbito de contextos de transacción existentes, debe configurar dicho servidor de aplicaciones web para que utilice el OTS de CORBA. Si esto no es posible, su servidor de aplicaciones web no es totalmente compatible con el soporte de CICS enterprise Java. (Si desea conocer un modo de utilizar la aplicación de ejemplo EJB Bank Account para probar si su

servidor de aplicaciones web es totalmente compatible con el soporte de CICS enterprise Java, consulte “Una nota sobre las transacciones distribuidas” en la página 305).

Para forzar a que WebSphere Application Server utilice el OTS de CORBA:

1. En la WebSphere Administration Console, seleccione el separador de valores de JVM.
2. Especifique lo siguiente en la sección "System Properties" (Propiedades del sistema):

   ```
   com.ibm.ejs.jts.ControlSet.interoperabilityOnly=true
   com.ibm.ejs.jts.ControlSet.nativeOnly=false
   ```

   Guarde los cambios.
3. Reinicie el servidor de aplicaciones.

org.omg.CORBA.OBJECT_NOT_EXIST
Esto se puede producir cuando un cliente encuentra una referencia a un bean en el servidor de nombres JNDI pero el bean ya no está instalado en CICS.

org.omg.CORBA.UNKNOWN
Hay muchas razones para esta excepción, incluidos errores en el código y errores en CICS. Consulte la salida de CICS para obtener más pistas sobre la causa del problema.

En muchos casos, la excepción CORBA incluye un código menor específico de CICS para ayudar en la determinación de problemas. Actualmente CICS utiliza los siguientes códigos menores:

Tabla 21. Códigos menores de CORBA específicos de CICS

Código	Componente de CICS que detecta el problema
1229111296	Receptor de peticiones de IIOP de CICS.
1229111297	Cualquier otro lugar del dominio de CICS II
1229111298	Componente ORB del dominio OT de CICS
1229111299	Componente JTS del dominio OT de CICS
1229111300	Componente CSI del dominio OT de CICS
1229111301	Componente CSI del dominio EJ de CICS

Si el cliente recibe una excepción CORBA que contenga cualquiera de los códigos menores de CICS, debería examinar los registros de mensajes de CICS para obtener más información sobre el error.

Problemas de versión de clase con RMI-IIOP

La invocación de método remoto sobre protocolo Internet InterORB (RMI-IIOP) es el protocolo de comunicación utilizado en CICS por enterprise beans y por objetos CORBA sin estado. Por lo tanto, la información de esta sección se aplica tanto a enterprise beans como a objetos CORBA sin estado.

Java RMI es un protocolo de objeto por valor. Esto significa que, siempre que se utiliza un objeto Java como parámetro en una llamada a método, lo que se envía por la red es el estado del objeto. Esto también sucede con los tipos de retorno y con las excepciones. Este estado es un objeto Java “serializado”. El estado lo puede deserializar la JVM remota para crear una nueva copia del objeto original en la JVM remota. El estado serializado contiene, entre otras cosas, un número de versión que indica la versión de la clase que representa el estado. Para que la JVM

remota pueda deserializar el objeto serializado, es necesario que esté presente la misma versión del archivo de clase en ambos extremos de la conexión IIOP. Si la JVM no comprende el estado del objeto, probablemente provocará que se genere la siguiente excepción:

```
java.rmi.MarshalException:unable to read from underlying bridge
```

(Esta excepción se puede emitir también por otras razones).

Si crea una clase en Java, puede incluir su propio mecanismo de serialización personalizado. Mediante este mecanismo, puede gestionar el mantenimiento de versiones de sus clases explícitamente, en lugar de depender del proceso de serialización predeterminado de Java. Además, si incluye un mecanismo de serialización personalizado, puede lograr ahorros de rendimiento importantes respecto al mecanismo predeterminado. Si desea aprovechar la serialización personalizada, sus objetos deben implementar la interfaz `java.io.Externalizable`.

A menudo, los objetos que se deben serializar son instancias de clases de la biblioteca de clases Java estándar. Por lo general, estas no cambian de una versión de Java a la siguiente, pero si lo hacen pueden dar origen al tipo de problema que se describe anteriormente. Para minimizar estos problemas, se recomienda utilizar la misma versión de Java en las máquinas asociadas que la que utilice CICS. Por ejemplo, entre Java 1.3.1 y Java 1.4, la clase `java.lang.Throwable` cambió significativamente. Esta clase es el supertipo de todas las excepciones en Java, por lo que muchas excepciones serializadas por Java 1.4.1 y versiones posteriores no se pueden deserializar mediante versiones anteriores.

En CORBA existe un mecanismo que utilizan muchos ORB para solventar el problema de los cambios de versión en las clases. Sin embargo, ese mecanismo no funciona perfectamente en CICS, ya que implica afinidades entre el ORB del socio y la JVM en CICS. Es probable que varias llamadas de RMI-IIOP al mismo objeto CORBA en CICS se procesen en distintas JVM. Esto hace que las afinidades que no estén soportadas y que el mecanismo para evitar los problemas de mantenimiento de versiones de clases no funcionen en CICS. Las aplicaciones CICS padecen este problema solo al enviar objetos serializados a una JVM remota. Si una JVM remota envía un objeto serializado a CICS, CICS puede utilizar el mecanismo CORBA estándar para tratar cualquier incompatibilidad de versión.

Si sufre este tipo de problema y no puede modificar la versión de Java que utiliza el socio en la plataforma, se recomienda modificar la aplicación para que utilice un tipo de datos que no provoque problemas de mantenimiento de versiones.

Utilización de mandatos de rastreo y de capacidad de servicio de EJB

Tal vez quiera rastrear una solicitud de EJB al intentar diagnosticar solicitudes que se cuelgan o que tienen anomalías, si tiene que poder identificar de forma exclusiva todas las transacciones asociadas con una única solicitud para supervisar esa actividad o, quizá, para fines de contabilidad.

Los problemas principales al intentar diagnosticar solicitudes que se cuelgan o con anomalías cuando un servidor lógico EJB cuenta con varias regiones CICS son que es necesario determinar:

- La región en la que se originó la solicitud (el receptor de peticiones).
- El destino (una región CICS u otro servidor) al que se ha dirigido la solicitud.

Los mandatos de la interfaz de programación del sistema (SPI) **INQUIRE WORKREQUEST** y **SET WORKREQUEST** le permiten:

- Determinar qué transacciones están asociadas con una única solicitud.
- Correlacionar todas las transacciones asociadas con una única solicitud.
- Depurar las solicitudes de trabajo seleccionadas.

Cada solicitud muestra:

- El número de tarea local y el ID de transacción.
- El tipo de solicitud; el primer tipo soportado es IIOP.
- Una serie exclusiva (imprimible) que se puede escribir en el mandato como filtro; por ejemplo:
  - Worktype
  - ClientIPAddress
  - Dirección TCP/IP o identificador de aplicación de SNA de destino (z/OS Communications Server).

Para obtener más información sobre estos mandatos, consulte los manuales *Referencia de programación del sistema CICS* y *Transacciones suministradas de CICS*.

Los mandatos INQUIRE y SET WORKREQUEST solo están disponibles para tareas de IIOP.

Las solicitudes de trabajo asociadas con receptores de solicitudes no es incluyen; son de tipo ligero, y toda esta información está disponible en el procesador de solicitudes. Un receptor de peticiones puede procesar más de una solicitud por instancia y puede haber abandonado el sistema mucho antes de que se haya completado la solicitud.

Cuando se interroga a un servidor lógico utilizando la WUI de CPSM, se tiene una única pantalla que visualiza todas las solicitudes de trabajo del servidor.

Con estos mandatos se puede depurar un procesador de solicitudes de forma parecida a como se depura una tarea de la lista CEMT INQ TASK.

Gestión de la seguridad para enterprise beans

El mecanismo de seguridad, la seguridad Java2, la capa de sockets seguros (SSL), la seguridad MRO y los roles de seguridad se pueden utilizar con enterprise beans.

Puede implementar cualquier combinación de los siguientes elementos.

Seguridad Java
Esta forma de control de la seguridad la implementa la máquina virtual Java (JVM) y se puede utilizar con cualquier programa Java que se ejecute bajo control de JVM. Consulte “Habilitación del gestor de seguridad Java” en la página 93 para obtener orientación sobre cómo configurar este tipo de control de seguridad.

Seguridad de capa de sockets seguros (SSL)
La capa de sockets seguros (SSL) es un protocolo de seguridad que proporciona privacidad y autenticación entre clientes y servidores que se comuniquen usando TCP/IP. Para obtener más información sobre SSL, consulte Soporte para protocolos de seguridad en la Guía de seguridad RACF.

Seguridad de MRO
Después de que el receptor de peticiones haya establecido un ID de usuario de

CICS que asociar con la solicitud, esta tal vez deba dirigirse a una región propietaria de la aplicación (AOR). Si el mecanismo de direccionamiento utiliza una conexión de operación multirregión (MRO), la transmisión del ID de usuario está sujeta a las reglas de seguridad de MRO. Consulte Seguridad de enlace con MRO.

Roles de seguridad
Un rol de seguridad representa un tipo de usuario de una aplicación en términos de los permisos que dicho usuario debe tener para utilizar correctamente la aplicación. Consulte “Roles de seguridad” en la página 367.

El archivo de política de enterprise beans proporcionado por CICS

El archivo de política de enterprise beans proporcionado por CICS, `dfjejbpl.policy`, se basa en el mecanismo de política de seguridad de Java.

El mecanismo de política de seguridad Java se describe en la *Enterprise JavaBeans Specification, versión 1.1*. El archivo de política de ejemplo se muestra en Figura 31 en la página 364.

En Java, la política de seguridad se define en términos de dominios de protección que correlacionan permisos con orígenes de código. Un dominio de protección contiene un origen de código con un conjunto de permisos asociados.

El archivo de política de enterprise beans proporcionado por CICS define dos dominios de protección, que hacen lo siguiente:

1. Otorga los permisos necesarios al origen de código del contenedor de enterprise beans de CICS para ejecución. Consulte el bloque 'grant codeBase' en Figura 31 en la página 364.
2. Otorga a cualquier origen de código únicamente los permisos descritos en la especificación *Enterprise JavaBeans* , versión 1. Consulte el bloque predeterminado 'grant' en Figura 31 en la página 364:
   - Para permitir que cualquiera lance una solicitud de trabajo de impresión.
   - Para permitir una conexión saliente en cualquier puerto TCP/IP.
   - Para permitir leer todas las propiedades del sistema.

Recuerde que, si desea utilizar JDBC o SQLJ desde enterprise beans, debe modificar el archivo de política de enterprise beans proporcionado por CICS para otorgar permisos al controlador JDBC. Para obtener más información, consulte Uso de JDBC y SQLJ para acceder a datos de DB2 desde programas Java en la Guía DB2.

```
// Permisos otorgados para dominio de protección de origen de código de contenedor
                de enterprise beans.
//de CICS
   grant codeBase "file:usr/lpp/cicsts/cicsts42//-" {
     permission java.security.AllPermission;
   };

// Permisos de EJB 1.1 predeterminados otorgados a todos los dominios de protección
   grant {
     // permite que cualquiera lance una solicitud de trabajo de impresión
     permission java.lang.RuntimePermission "queuePrintJob";

     // permite una conexión saliente en cualquier puerto TCP/IP
     permission java.net.SocketPermission "*:0-65535", "connect";

     // permite que cualquiera lea propiedades
     permission java.util.PropertyPermission "*", "read";
   };
```

Figura 31. Política de seguridad de enterprise beans de CICS de ejemplo

Utilización de seguridad de enterprise beans

La especificación EJB 1.1 define las siguientes API de seguridad para permitir que los enterprise beans tomen decisiones de aplicación basas en detalles de seguridad del interlocutor.

java.security.Principal getCallerPrincipal()
Este método se utiliza para determinar quién invocó el método del bean actual. El método getCallerPrincipal está soportado completamente en CICS. Los detalles sobre cómo se determina la identidad del interlocutor actual se muestran en "Derivación de nombres distinguidos" en la página 366.

boolean isCallerInRole(String SecurityRoleReference)
Este método se utiliza para probar si el interlocutor actual está asignado a un rol de seguridad que está enlazado a la referencia de rol de seguridad especificada en la llamada a método.

CICS generará una excepción de tiempo de ejecución (que es conforme a la especificación EJB 1.1) si se utilizan las siguientes API de seguridad de EJB 1.0 en desuso.

- java.security.Identity getCallerIdentity()
- boolean isCallerInRole(java.security.Identity role)

Nota: Los enterprise beans desarrollados según la especificación de Enterprise JavaBeans (EJB) 1.0 tienen que actualizarse al nivel de la especificación Enterprise JavaBeans 1.1 utilizando las herramientas de desarrollo que se proporcionan.

- Consulte "Las herramientas de despliegue para enterprise beans en un sistema CICS" en la página 321 para obtener información sobre las herramientas de despliegue.
- Consulte "Escritura de enterprise beans" en la página 307 para obtener información sobre la escritura de enterprise beans.

Definición de los permisos de acceso a archivos para enterprise beans:

Para ejecutar enterprise beans correctamente en CICS, el ID de usuario de la región CICS debe tener permiso para acceder a los archivos utilizados por la lógica de la empresa.

Estos permisos de archivo son necesarios para ejecutar enterprise beans, independientemente del nivel de seguridad implementado. Consulte también la *Guía de instalación de CICS Transaction Server para z/OS*.

Acceso a archivos z/OS UNIX utilizados por enterprise beans:

Estos permisos de archivo son necesarios para ejecutar enterprise beans.

Tabla 22. Permisos de acceso a archivos necesarios para enterprise beans de CICS

Estructura de archivos/directorios	Permiso mínimo	Comentarios
Directorio de estantería CORBASERVER (por ejemplo, `/var/cicsts/`)	Lectura, grabación y ejecución	Se accede al directorio de estantería durante la instalación de CORBASERVER y DJAR, y cada CICS tiene que crear subdirectorios exclusivos (consulte la nota 1).
Estructura del directorio `/usr/lpp/cicsts/cicsts42` y clases	Lectura y ejecución	Contiene el código Java proporcionado por CICS (consulte la nota 2).
Directorios `/usr/lpp/java/J6.0.1_64/bin` y `/usr/lpp/java/J6.0.1_64/bin/classic`.	Lectura y ejecución	Contiene el código de JVM de IBM (consulte la nota 3).
Directorio de trabajo de CICS	Lectura, grabación y ejecución	Se utiliza para crear archivos `stdin` (consulte la nota 4).
Archivo JAR desplegado	Lectura	El proceso de despliegue los utiliza durante la instalación de DJAR.
Archivo de política de seguridad (si es necesario)	Lectura	Necesario si la propiedad **`-Djava.security.policy`** se especifica en el archivo de propiedades del sistema de la JVM.
Archivo de propiedades del sistema	Lectura	Opcional al crear una JVM (consulte la nota 5).

Nota:

1. `/var/cicsts/` es el nombre predeterminado del directorio de estantería (SHELF) cuando se define una definición de recurso CORBASERVER. Cada región CICS crea un subdirectorio exclusivo en esta estantería cuando instala la definición de recurso.
2. *cicsts42* es el valor elegido para el parámetro de instalación USSDIR que definiera al instalar CICS TS.
3. `java/J6.0.1_64` es la ubicación de instalación para el IBM 64 bits SDK para z/OS, Java Technology Edition.
4. El directorio de trabajo de CICS lo define el parámetro WORK_DIR en el perfil de JVM.
5. El directorio y el nombre de archivo opcionales de propiedades del sistema se indican en la opción JVMPROPS del perfil de JVM.

La propiedad de los archivos y los permisos pueden definirse utilizando los mandatos **chmod** y **chown**. Para obtener más información, consulte la publicación *z/OS UNIX System Services: Command Reference*.

Acceso a conjuntos de datos utilizados por enterprise beans:

Antes de poder instalar cualquier CORBASERVER en una región CICS, es necesario crear, definir para CICS e instalar los dos siguientes conjuntos de datos

con acceso UPDATE (actualización). Estos archivos pueden ser conjuntos de datos VSAM o tablas de datos de recurso de acoplamiento.

La Figura 32 muestra un ejemplo de mandatos de RACF para acceder a conjuntos de datos con la autorización necesaria.

Nota: Estos archivos los utiliza CICS internamente, por lo que ningún usuario debería recibir acceso de seguridad de nivel de recurso para acceder a ellos. Esto impedirá que aplicaciones VSAM accedan a los datos de estos archivos.

DFHEJDIR
Este conjunto de datos contiene un directorio de secuencias de solicitudes que comparten las regiones de escucha y las AOR que conforman un servidor IIOP de CICS. El archivo debe ser recuperable.

DFHEJOS
DFHEJOS es un conjunto de datos que contiene beans de sesión con estado desactivados. Lo comparten todas las AOR que forman parte de un servidor IIOP de CICS. Este archivo no debe ser recuperable.

```
ADDSD  'CICSTS42.CICS.CICS.DFHEJDIR' NOTIFY(id_admin_sist_cics) UACC(NONE)
PERMIT 'CICSTS42.CICS.CICS.DFHEJDIR' ID(id_cics1,...,grupo_cics1,..,grupo_cicsn)
        ACCESS(UPDATE)
ADDSD  'CICSTS42.CICS.CICS.DFHEJOS' NOTIFY(id_admin_sist_cics) UACC(NONE)
PERMIT 'CICSTS42.CICS.CICS.DFHEJOS' ID(id_cics1,...,grupo_cics1,..,grupo_cicsn)
        ACCESS(UPDATE)
```

Figura 32. Ejemplo de mandatos de RACF para autorizar el acceso a conjuntos de datos de CICS

Consulte Authorizing access to CICS data sets, en la *Guía de seguridad RACF de CICS*, para obtener más información sobre cómo autorizar el acceso a conjuntos de datos de CICS.

Derivación de nombres distinguidos:

Los enterprise beans pueden identificar su usuario final o su cliente mediante un objeto *Principal*.

El método getCallerPrincipal devuelve un objeto Principal que representa al cliente, y dicho objeto Principal contiene métodos que se pueden invocar para devolver información sobre dicho cliente. En concreto, el método getName del objeto Principal devuelve una serie que contiene el "nombre distinguido" del cliente. El nombre distinguido o DN es una secuencia de pares de palabra clave y valor, conocidos como nombres distinguidos relativos o RDN, y forma parte de la recomendación X.500 (estándar ISO/IEC 9594). La representación de serie de un nombre distinguido la sugiere RFC2253, *LDAP V3: UTF-8 String Representation of Distinguished Names* (representación de serie de nombres distinguidos).

Nota: CICS Transaction Server para z/OS, Versión 4 Release 2 no verifica que una instancia de bean de sesión con estado lo utiliza únicamente el mismo principal que la creó. Por lo tanto, el ID de usuario y el nombre distinguido del principal pueden ser distintos tras la reactivación de una instancia de bean.

Si se ha identificado y autenticado el cliente del bean mediante un certificado de cliente utilizando el protocolo de capa de socket seguros, el nombre distinguido siempre se obtiene de dicho certificado. Sin embargo, si el cliente del bean no ha proporcionado ningún certificado, el nombre distinguido se obtiene invocando el módulo sustituible por el usuario DFHEJDNX. Las entradas para el módulo

DFHEJDNX son el título, la unidad organizativa, la organización, la localidad, el estado y el país, que se obtienen del certificado del servidor cuyo código se especifica en la opción CERTIFICATE de la definición CORBASERVER, así como el ID de usuario y el nombre común asociados con el ID de usuario del usuario que ejecuta el bean, pero si se especifica SEC=NO, se utiliza el ID de usuario de la región CICS. El nombre común se deriva mediante la transformación del nombre de usuario para ese usuario en una serie con mayúsculas y minúsculas. El código del certificado especifica un certificado con el conjunto de claves identificado mediante el parámetro de inicialización del sistema KEYRING. Si se omite la opción CERTIFICATE, la información se obtiene del certificado predeterminado del conjunto de claves. Si se omite el parámetro KEYRING, no se pasa ninguna información de certificado a DFHEJDNX y solo está disponible el RDN del nombre común.

La versión proporcionada por CICS de DFHEJDNX acepta las entradas derivadas del certificado y del nombre de usuario de CORBASERVER y le da el formato de un nombre distinguido con el siguiente estilo:

T=CICS EJB Container,CN=Louise Peters,OU=CICS/390 Development, O=IBM,L=Hursley,ST=Hampshire,C=GB

Los ejemplos de DFHEJDNX proporcionados por CICS están ubicados en la biblioteca SDFHSAMP, *CICSTS42.CICS.CICS.SDFHSAMP*, de este modo:

- DFHEJDN1 para lenguaje ensamblador.
- DFHEJDN2 para lenguaje C.

Roles de seguridad

El acceso a métodos de enterprise bean se basa en el concepto de *roles de seguridad*. Un rol de seguridad representa un tipo de usuario de una aplicación en términos de los permisos que dicho usuario debe tener para utilizar correctamente esta aplicación.

Por ejemplo, en la aplicación de nóminas:

- Un rol de `gestor` podría representar a los usuarios que tienen permiso para utilizar todas las partes de la aplicación.
- Un rol de `líder_de_equipo` podría representar a los usuarios que tienen permiso para utilizar las funciones de administración de la aplicación.
- Un rol de `entrada_de_datos` podría representar a los usuarios que tienen permiso para utilizar las funciones de entrada de datos de la aplicación.

Los roles de seguridad correspondientes a una aplicación los define el ensamblador de la aplicación y se especifican en el descriptor de despliegue del bean. Para obtener más información, consulte “Roles de seguridad en el descriptor de despliegue” en la página 372

Los roles de seguridad que tienen permiso para ejecutar un método de bean también se especifican en el descriptor de despliegue del bean, y esto también lo hace el ensamblador de la aplicación. En el ejemplo, los métodos que actualizan las horas trabajadas por los empleados cada semana se pueden asignar al rol `entrada_de_datos`, mientras que los métodos que suprimen a un empleado de la nómina se pueden asignar al rol `líder_de_equipo`.

Para distinguir roles de seguridad con nombres similares en aplicaciones distintas o en sistemas distintos, los roles de seguridad especificados en el descriptor de

despliegue del bean pueden recibir un calificador de una o de dos partes cuando el bean se despliega en un sistema CICS. Por ejemplo:

- Rol de seguridad si ningún calificador:

  `líder_de_equipo`
- Rol de seguridad con un calificador:

  `nómina.líder_de_equipo`
- Rol de seguridad con dos calificadores:

  `prueba.nómina.líder_de_equipo`

Un rol de seguridad con sus calificadores se conoce como un **rol de seguridad desplegado**. Para obtener más información, consulte "Roles de seguridad desplegados".

La correlación de roles de seguridad con usuarios individuales se realiza en el gestor de seguridad externa. La correlación no es necesariamente de uno a uno. Por ejemplo, se pueden asignar varios usuarios al rol `entrada_de_datos`, mientras algunos usuarios se pueden asignar al rol `líder_de_equipo` y al rol `entrada_de_datos`. Para obtener más información, consulte "Implementación de roles de seguridad" en la página 374.

El rol de seguridad y el nombre de visualización en el descriptor de despliegue pueden contener cualquier carácter ASCII o Unicode. Esto no es así para los nombres que se utilizan en RACF, que están restringidos a los caracteres de la página de códigos EBCDIC 037. Además, algunos caracteres —por ejemplo, el asterisco (`*`)— tienen un significado especial cuando se utilizan en mandatos de RACF. Por lo tanto, cuando CICS construye el rol de seguridad desplegado desde sus componentes, algunos caracteres se sustituyen por un carácter distinto y otros se sustituyen por una secuencia de escape. Para obtener más detalles, consulte "Sustitución de caracteres en roles de seguridad desplegados" en la página 370.

Roles de seguridad desplegados:

Una correlación directa entre los roles de seguridad especificados en el descriptor de despliegue de un bean y usuarios individuales puede no controlar adecuadamente el acceso a métodos de bean.

Por ejemplo:

- Dos aplicaciones, de distintos proveedores, pueden utilizar nombres similares para roles de seguridad. En su empresa, los usuarios de cada aplicación pueden ser distintos.
- Un bean se podría utilizar en más de una aplicación. Un usuario puede estar autorizado a utilizar un método concreto en una aplicación pero no en la otra.
- Una aplicación se podría desplegar en un sistema de prueba y en un sistema de producción. Los miembros del departamento de pruebas pueden tener permiso para utilizar todos los métodos de bean en el sistema de prueba, pero no en el sistema de producción.

Para proporcionar el grado de control necesario en estos y en otros casos, puede calificar los roles de seguridad en el nivel de aplicación y en el nivel de sistema. Un rol de seguridad con sus calificadores se conoce como un **rol de seguridad desplegado**. A continuación hay un ejemplo de un nombre de rol cualificado en ambos niveles:

`prueba.nómina.líder_de_equipo`

- `nómina` califica el rol de seguridad en el nivel de la aplicación y se utiliza para distinguir entre el rol `líder_de_equipo` en la aplicación de nóminas y el rol `líder_de_equipo` en otras aplicaciones.
- `prueba` califica el rol de seguridad en el nivel del sistema y se utiliza para distinguir entre el rol nómina.líder_de_equipo en el sistema de prueba y el rol nómina.líder_de_equipo en otros sistemas.

En el nivel de aplicación, los roles de seguridad se califican mediante el **nombre de visualización**, si se especifica alguno en el descriptor de despliegue. Si no se especifica ningún nombre de visualización, los roles de seguridad no se califican en el nivel de aplicación. Si se utiliza un calificador de nivel de aplicación, se emplea un punto (`.`) como delimitador; si no se utiliza ningún calificador, no hay delimitador.

En el nivel de sistema, opcionalmente los roles de seguridad se califican mediante un prefijo que se especifica en el parámetro de inicialización del sistema EJBROLEPRFX. Si no se especifica EJBROLEPRFX, los roles de seguridad no se califican en el nivel de sistema. Si se utiliza un calificador de nivel de sistema, se emplea un punto (`.`) como delimitador; si no se utiliza ningún calificador, no hay delimitador.

Este ejemplo muestra cómo se pueden calificar los roles de seguridad definidos en el descriptor de despliegue de un bean:

- Un bean contiene tres roles de seguridad: gestor, líder_de_equipo y entrada_de_datos
- El bean se utiliza en una aplicación de nómina, con un nombre de visualización de `nómina`. El bean también forma parte de una aplicación de prueba, que no cuenta con nombre de visualización.
- La aplicación de nómina se utiliza en dos sistemas de producción: el primero no especifica ningún prefijo, mientras que el segundo especifica un prefijo de `ejecutivo`.
- La aplicación de prueba se utiliza en un sistema de prueba con un prefijo de `prueba1`.

Cuando se aplican los dos niveles de cualificación a los roles de seguridad especificados en el descriptor de despliegue, los roles de seguridad desplegados son:

```
nómina.gestor        ejecutivo.nómina.gestor        prueba1.gestor
nómina.líder_equipo    ejecutivo.nómina.líder_equipo    prueba1.líder_equipo
nómina.entrada_de_datos     ejecutivo.nómina.entrada_de_datos     prueba1.entrada_de_datos
```

Cada uno de estos roles desplegados se puede correlacionar con usuarios individuales (o con grupos de usuarios) para ajustar la seguridad necesaria para la empresa.

Si un rol de seguridad no está cualificado en el nivel de aplicación, o en el nivel de sistema, el rol de seguridad desplegado es el mismo que el rol de seguridad definido en el descriptor de despliegue. Por ejemplo, si el bean del ejemplo anterior se utiliza en una aplicación que no cuente con nombre de visualización y la aplicación se utiliza en un sistema que no especifique EJBROLEPRFX, los roles de seguridad desplegados son:

```
gestor
líder_de_equipo
entrada_de_datos
```

Habilitación e inhabilitación del soporte de roles de seguridad:

De forma predeterminada, el soporte de roles de seguridad de CICS está habilitado.

Puede utilizar el parámetro de inicialización del sistema XEJB para inhabilitar (o habilitar explícitamente) el soporte de roles de seguridad. Si inhabilita el soporte:

- CICS no realiza comprobaciones de autorización de método: todos los usuarios tienen permiso para utilizar todos los métodos de bean.
- El método isCallerInRole() devuelve `true` para todos los usuarios.

Referencias de rol de seguridad:

En una aplicación, se puede utilizar el método isCallerInRole() para determinar si el usuario de la aplicación tiene definido un rol determinado.

El método toma un **referencia de rol de seguridad** como un argumento y no como un rol de seguridad. Las referencias de rol de seguridad codificadas en el bean las define el proveedor y se declaran en el descriptor de despliegue del bean.

Para obtener más información, consulte “Roles de seguridad en el descriptor de despliegue” en la página 372

El ensamblador de la aplicación enlaza cada referencia de rol de seguridad con un rol de seguridad; el enlace se declara en el descriptor de despliegue correspondiente al bean. Por ejemplo, la referencia de rol de seguridad de administrador utilizada en el código del bean puede enlazarse, en el descriptor de despliegue, con el rol líder_de_equipo.

Para obtener más información, consulte “Roles de seguridad en el descriptor de despliegue” en la página 372

Sustitución de caracteres en roles de seguridad desplegados:

El rol de seguridad y el nombre de visualización que hay en el descriptor de despliegue pueden contener cualquier carácter ASCII o Unicode.

El juego de caracteres que se puede utilizar en roles de seguridad desplegados está más restringido:

- Los nombres de perfil utilizados en RACF están restringidos a caracteres de la página de códigos EBCDIC 037.
- Algunos caracteres —por ejemplo, el asterisco (*)— tienen un significado especial cuando se utilizan en mandatos de RACF y no se pueden utilizar en un nombre de perfil.

Cuando los caracteres Unicode del rol de seguridad y del nombre de visualización no se pueden utilizar directamente en el rol de seguridad desplegado, se sustituyen por las secuencias de escape mostradas en Tabla 23 en la página 371. La sustitución se produce:

- Cuando el programa de utilidad del generador EJBROLE (`dfhreg`) procesa el descriptor de despliegue para generar mandatos de RACF.
- Cuando CICS correlaciona un rol de seguridad con un ID de usuario de RACF.

Tabla 23. Secuencias de escape utilizadas en roles de seguridad

Carácter	Descripción	ASCII/Unicode	Página de códigos EBCDIC 037	Secuencia de escape
Los valores ASCII y Unicode cuyo valor EBCDIC equivalente no se pueda utilizar en el nombre de un rol de seguridad desplegado se sustituyen por una secuencia de escape de tres caracteres del siguiente modo:				
	en blanco	X'20'	X'40'	¢
¢	centavo	X'A2'	X'4A'	\A2
\	barra inclinada invertida	X'5C'	X'E0'	\5C
*	asterisco	X'2A'	X'5C'	\2A
&	carácter &	X'26'	X'50'	\26
%	porcentaje	X'25'	X'6C'	\25
,	coma	X'2C'	X'6B'	\2C
(	paréntesis izquierdo	X'28'	X'4D'	\28
)	paréntesis derecho	X'29'	X'5D'	\29
;	punto y coma	X'3B'	X'5E'	\3B
Los valores Unicode que no tienen un equivalente en la página de códigos EBCDIC 037 se sustituyen por la secuencia de escape Unicode: un carácter con una representación Unicode de X'yyyy' se sustituye por \uyyyy. Por ejemplo:				
€	símbolo del euro	X'20AC'	no está soportado	\u20AC
	Hiragana Ki	X'304D'	no está soportado	\u304D
α	alfa	X'03B1'	no está soportado	\u03B1

A continuación hay dos ejemplos que ilustran el modo en el que se sustituyen los caracteres:

Ejemplo 1

- EJBROLEPRFX tiene un valor de `test`
- El nombre de visualización en el descriptor de despliegue tiene un valor de `year.end.processing`
- El rol de seguridad en el descriptor de despliegue tiene un valor de `auditor 1`

En este ejemplo, cuando se construye el rol de seguridad:

1. Cada espacio se sustituye por ¢
2. El rol de seguridad desplegado está compuesto por el valor EJBROLEPRFX, el nombre de visualización y el rol de seguridad; como delimitador se utiliza el punto.

El rol de seguridad desplegado resultante es:

```
test.year.end.processing.auditor¢1
```

Ejemplo 2

- EJBROLEPRFX tiene un valor de `test`

- El nombre de visualización en el descriptor de despliegue tiene un valor de αβ32 La codificación Unicode es X'03B1 03B2 0033 0034'.
- El rol de seguridad en el descriptor de despliegue tiene un valor de `auditor 1`

En este ejemplo, cuando se construye el rol de seguridad:

1. Cada carácter Unicode que tiene un equivalente en la página de códigos EBCDIC 037 se sustituye correspondientemente: en el nombre de visualización, X'0033 0034' se sustituye por 34.
2. Cada carácter Unicode que ***no*** tiene ningún equivalente en la página de códigos EBCDIC 037 se sustituye por la secuencia de escape correspondiente. En el nombre de visualización, X'03B1 03B2' se sustituye por `\u03B1\u03B2`
3. Cada espacio se sustituye por `¢`
4. El rol de seguridad desplegado está compuesto por el valor EJBROLEPRFX, el nombre de visualización y el rol de seguridad; como delimitador se utiliza el punto.

El rol de seguridad desplegado resultante es:

```
test.\u03B1\u03B234.auditor¢1
```

Roles de seguridad en el descriptor de despliegue:

Aquí se muestra un fragmento de un descriptor de despliegue. Incluye la siguiente información sobre roles de seguridad.

- 1 Un nombre de visualización de `nómina`.
- 2 La referencia de rol de seguridad de `administrador` que se enlaza con el rol `líder_equipo`.
- 3 Un rol de seguridad de `líder_equipo`.
- 4 Un permiso de método que permite a un usuario definido en el rol `líder_equipo` invocar el método create().

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ejb-jar PUBLIC
"-//Sun Microsystems, Inc.//DTD Enterprise JavaBeans 1.1//ES"
"http://java.sun.com/j2ee/dtds/ejb-jar_1_1.dtd">
  <ejb-jar id="ejb-jar_ID">
    <display-name>nómina</display-name>         1
      <enterprise-beans>
        <session id="Session_1">
          .
          .
          <security-role-ref id="SecurityRoleRef_1">
            <role-name>administrador</role-name> 2
            <role-link>líder_equipo</role-link>
          </security-role-ref>
          .
          .
        </session>
      </enterprise-beans>
      <assembly-descriptor id="AssemblyDescriptor_1">
        <security-role id="SecurityRole_1">
          <role-name>líder_equipo</role-name>     3
        </security-role>
        .
        .
        <method-permission id="MethodPermission_1">
          <description>líder_equipo:+:</description>
          <role-name>líder_equipo</role-name>     4
          <method id="MethodElement_01">
            <ejb-name>Managed</ejb-name>
            <method-intf>Home</method-intf>
            <method-name>create</method-name>
            <method-params>
            </method-params>
          </method>
           .
           .
        </method-permission>
         .
         .
      </assembly-descriptor>
       .
       .
     </ejb-jar>
```

Figura 33. Ejemplo de un descriptor de despliegue que contiene roles de seguridad

Si se utiliza una aplicación con este descriptor de despliegue en un sistema CICS con los siguientes parámetros de inicialización del sistema:

```
SEC=YES
XEJB=YES
EJBROLEPRFX='test'
```

- El rol de seguridad desplegado de `prueba.nómina.líder_equipo` debe definirse para RACF.
- Los usuarios con acceso de lectura (READ) a ese rol de seguridad desplegado tienen permiso para invocar el método create().
- `isCallerInRole('administrator')` devolverá `true` para usuarios definidos en el rol de seguridad desplegado de `prueba.nómina.líder_equipo` y `false` para los demás usuarios.

Para obtener información detallada sobre el contenido del descriptor de despliegue, consulte la *Enterprise JavaBeans Specification, versión 1.1*.

Para visualizar el contenido de un descriptor de despliegue, puede utilizar el Assembly Toolkit (ATK). Para obtener más información sobre ATK, consulte The enterprise bean deployment tool, ATK, en la *Guía de operaciones y programas de utilidad de CICS*.

Implementación de roles de seguridad

El acceso a métodos de enterprise bean se basa en el concepto de **roles de seguridad**.

Acerca de esta tarea

Estos se describen en "Roles de seguridad" en la página 367.

Para implementar la utilización de roles de seguridad en un entorno de enterprise bean de CICS, debe hacer los siguiente:

1. Determine qué roles de seguridad se definen en el descriptor de despliegue de la aplicación.
2. Determine los nombres de visualización asociados con los roles de seguridad en el descriptor de despliegue de la aplicación. El nombre de visualización califica el rol de seguridad en el nivel de la aplicación.
3. Decida si es necesario calificar el nombre del rol de seguridad en el nivel del sistema y, si es así, el valor del prefijo que utilizará en cada sistema en el que se ejecute la aplicación.
4. Mediante la información recopilada en los pasos del 1 al 3, determine los nombres de los roles de seguridad desplegados que utiliza la aplicación en cada sistema. Los caracteres del rol de seguridad y del nombre de visualización que no tengan un equivalente directo en la página de códigos EBCDIC 37 (así como algunos otros caracteres) deben sustituirse por un carácter distinto o por una secuencia de escape al construir el rol de seguridad desplegado. Consulte "Sustitución de caracteres en roles de seguridad desplegados" en la página 370 para obtener más información.
5. Mediante la información recopilada en los pasos del 1 al 3, defina los perfiles de RACF para los roles de seguridad desplegados. Consulte "Definición de roles de seguridad para RACF" en la página 375 para obtener más información.
6. Asocie usuarios individuales o grupos de usuarios con cada rol de seguridad desplegado en RACF. Consulte "Definición de roles de seguridad para RACF" en la página 375 para obtener más información.
7. Especifique los siguientes parámetros de inicialización del sistema:
   - `SEC=YES`
   - `XEJB=YES`. Este es el valor predeterminado, por lo que no es necesario especificarlo de forma explícita.
8. Para aquellos sistemas en los que los roles de seguridad desplegados contengan un calificador de nivel de sistema (consulte el paso 3), especifique el parámetro de inicialización del sistema EJBROLEPRFXEJBROLEPRFX.

Utilización del programa de utilidad generador EJBROLE de RACF:

El programa de utilidad generador EJBROLE de RACF, dfhreg, es un programa Java que extrae información sobre roles de seguridad de descriptores de despliegue y genera un programa REXX que se puede utilizar para definir los roles de seguridad para RACF.

El programa REXX que dfhreg genera contiene los mandatos de RACF que definen los roles de seguridad como miembros de un perfil en la clase GEJBROLE. Antes de ejecutar el programa REXX, modifíquelo para cambiar el nombre del perfil que se define.

Los scripts de invocación de dfhreg para z/OS UNIX (`dfhreg`) y para Windows (`dfhreg.bat`) están en el directorio `$CICS_HOME/lib/security`. La implementación de dfhreg (`dfhreg.jar`) también se encuentra en este directorio. Los demás archivos JAR necesarios para ejecutar dfhreg (`dfjcsi.jar`, `dfjejbdd.jar` y `dfjorb.jar`) están en el directorio `$CICS_HOME/lib`. `$CICS_HOME` es el directorio de z/OS UNIX en el que ha instalado los componentes USS de CICS.

Puede ejecutar dfhreg en cualquier plataforma que soporte Java; sin embargo, debe ejecutar el programa REXX resultante para la base de datos RACF en el sistema z/OS en el que desee definir los roles de seguridad. Cuando se ejecuta dfhreg, debe cumplir los siguientes requisitos:

1. La vía de acceso de clases contiene los siguientes archivos JAR:

   ```
   dfhreg.jar
   dfjcsi.jar
   dfjejbdd.jar
   dfjorb.jar
   ```

2. Debe estar utilizando una versión soportada del Java 2 SDK.

El programa REXX que el programa de utilidad genera está en la página de códigos de la plataforma en la que se ejecuta el programa de utilidad. Si ejecuta el programa de utilidad en una plataforma que utilice una página de códigos ASCII, debe convertir el programa REXX a la página de códigos EBCDIC que se utiliza en el sistema z/OS de destino.

Definición de roles de seguridad para RACF:

En RACF, los roles de seguridad desplegados se gestionan como recursos generales. Para definir los roles de seguridad desplegados, defina perfiles en las clases de recursos GEJBROLE o EJBROLE, con las listas de acceso adecuadas.

Por ejemplo, para utilizar los siguientes mandatos para definir los roles de seguridad desplegados `rol_seguridad_desplegado_1` y `rol_seguridad_desplegado_2` como miembros del perfil `grupo_rol_seguridad` de la clase GEJBROLE, y conceder acceso de lectura (READ) a `usuario1` y `usuario2`:

```
RDEFINE GEJBROLE grupo_rol_seguridad UACC(NONE)
                 ADDMEM(rol_seguridad_desplegado_1, rol_seguridad_desplegado_2, ...)
                 NOTIFY(id_usuario_admin)
PERMIT grupo_rol_seguridad CLASS(GEJBROLE) ID(usuario1, usuario2) ACCESS(READ)
```

Como alternativa, utilice los siguientes mandatos para definir los roles de seguridad desplegados en la clase EJBROLE y conceder a los usuarios acceso de lectura (READ) a cada rol de seguridad desplegado:

```
RDEFINE EJBROLE (rol_seguridad_desplegado_1, rol_seguridad_desplegado_2, ...) UACC(NONE)
                 NOTIFY(id_usuario_admin)
PERMIT rol_seguridad_desplegado_1 CLASS(EJBROLE) ID(usuario1, usuario2) ACCESS(READ)
PERMIT rol_seguridad_desplegado_2 CLASS(EJBROLE) ID(usuario1, usuario2) ACCESS(READ)
```

Nota:

1. El rol de seguridad que se especifica es el rol de seguridad desplegado y no el rol de seguridad sin calificar que se define en el descriptor de despliegue.

2. Para ejecutar un método de bean o para recibir una respuesta `true` del método isCallerInRole(), un usuario necesita acceso de lectura (READ).

CICSPlex SM con enterprise beans

La gestión de enterprise beans puede realizarse en todo un CICSplex.

Soporte de CICSPlex SM para enterprise beans

La gestión de enterprise beans puede realizarse en todo un CICSplex empleando los servicios Operator y API de CICSPlex SM.

La función que proporciona CICSPlex SM para el soporte de Enterprise JavaBeans incluye:

- Gestión de objetos para definiciones de CorbaServer y DJAR.
- Gestión de objetos para instancias instaladas de CorbaServer y DJAR.
- Gestión dinámica de la ejecución de enterprise beans.

Las áreas de CICSPlex SM que abarcan estos recursos son:

- La interfaz de programación de aplicaciones (API): para permitir la definición, la consulta y la gestión de objetos de enterprise bean mediante la interfaz de **EXEC CPSM**. Consulte la *Guía de programación de la aplicación de CICSPlex System Manager* para obtener más información.
- La interfaz de usuario web: para permitir la consulta y la gestión de objetos de enterprise bean mediante un navegador web. Consulte *Guía de la interfaz de usuario web de CICSPlex System Manager* para obtener información sobre la interfaz de usuario web.

Soporte de definiciones de CICSPlex SM para enterprise beans

Business Application Services (BAS) es el componente de CPSM relativo a la definición y la instalación de recursos de CICS.

Para obtener más información sobre BAS, consulte el manual *Aplicaciones de negocio de gestión de CICSPlex System Manager*. Los objetos de BAS específicos para Enterprise JavaBeans son:

- EJCODEF: definición de CorbaServer para enterprise bean
- EJDJDEF: definición de archivo JAR desplegado por CICS para enterprise bean

El objeto de definición de CorbaServer (EJCODEF) permite la especificación de exactamente las mismas característica de CorbaServer que la versión CEDA. EJCODEF se describe en Defining CorbaServers using BAS, en el manual *Aplicaciones de negocio de gestión de CICSPlex System Manager*.

El objeto de definición de archivo JAR desplegado por CICS (EJDJDEF) permite la especificación de exactamente las mismas característica de DJAR que la versión CEDA. EJDJDEF se describe en Defining a CICS-deployed JAR file using BAS, en el manual *Aplicaciones de negocio de gestión de CICSPlex System Manager*.

Estos recursos están completamente integrados en la funcionalidad estándar de BAS y se pueden gestionar e instalar de forma automática o cuando se desee, según necesite el usuario.

Además de estos dos tipos de objeto, hay otros objetos de BAS que están relacionados con la operación de los enterprise beans:

- TCPDEF: definición de TCPIPSERVICE.
- RQMDEF: definición de REQUESTMODEL.

- TRANDEF: definición de CICS TRANSACTION.
- PROGDEF: definición de PROGRAM.

Las solicitudes de ejecución de enterprise beans desde clientes llegan a la región de escucha de CICS mediante un puerto TCP/IP. Si se utiliza BAS, el número de este puerto se debe especificar mediante un objeto TCPDEF, que se debería instalar en todas las regiones de escucha que se espere que respondan a estas llamadas. El contenido de un TCPDEF debería duplicar el especificado para la definición TCPIPSERVICE de CEDA. Consulte “Configuración de TCP/IP para IIOP” en la página 407 para obtener información.

Si los usuarios necesitan que las solicitudes de ejecución para enterprise beans específicos sean reconocidas y gestionadas de forma distinta a las de las ejecuciones de enterprise beans genéricos, puede utilizarse un modelo de solicitud para asociarlo con un código de transacción especificado por el usuario. En CICSPlex SM, los modelos de solicitud se definen mediante objetos RQMDEF y se deben instalar en todas las regiones de escucha en las que sea necesario interceptar dichas solicitudes. En función de la complejidad del enterprise bean, tal vez sea necesario instalar además los modelos de solicitud en las AOR asociadas. El contenido de estos RQMDEF debería duplicar el especificado para la definición REQUESTMODEL de CEDA. Consulte “Obtención de un TRANSID de CICS” en la página 418 para obtener información.

En un entorno de proceso distribuido de enterprise beans, se esperaría que algunas regiones CICS actuaran como regiones de escucha para recibir las solicitudes de ejecución de IIOP y otras actuaran como las AOR, para brindar el entorno de EJB real para la ejecución de los enterprise beans necesarios. El objeto TRANDEF de CICSPlex SM es una herramienta especialmente potente para emplear en este caso, ya que se puede instalar un único objeto de definición de transacción tanto dinámicamente en las regiones de escucha como de forma estática en las AOR mediante una única asignación de recursos de BAS (RASGNDEF), como se describe en Resource assignments, en el manual *Aplicaciones de negocio de gestión de CICSPlex System Manager*.

Consideraciones sobre el ámbito lógico de BAS

Uno de los beneficios de utilizar BAS para definir e instalar suites de aplicaciones empresariales del usuario es que los usuario pueden definir el ámbito de sus vistas de objetos a los recursos pertinentes para sus instancias de aplicación instaladas.

Por ejemplo, si una aplicación empresarial consta de un conjunto concreto de archivos, transacciones y programas, las vistas LOCTRAN, LOCFILE y PROGRAM estarán reservadas exclusivamente para las vistas de instancias de objetos coincidentes en las regiones en las que se instalaron. El recurso para permitir esta vista de objetos restringida se conoce como "ámbito lógico". Los objetos CorbaServer y DJAR pueden participar en el ámbito lógico exactamente del mismo modo que otras definiciones tradicionales de BAS.

Nota: Los enterprise beans no se definen para CICS como tales. Se identifican ante CICS cuando sus DJAR asociados entran en servicio tras su instalación en una región CICS. Por lo tanto, los enterprise beans pueden "adoptar" un ámbito lógico mediante la asociación de sus DJAR. Sin embargo, la especificación Enterprise JavaBean permite que se instalen enterprise beans para aplicaciones diferentes en un único DJAR. Si se sigue esta práctica, al proceso de ámbito lógico le resultará imposible diferenciar entre los enterprise beans instalados y los nombres de aplicación empresarial adecuados. Por tanto, si los usuarios quieren aprovechar el ámbito lógico de BAS para aumentar sus vistas de CICSPlex de objetos de

enterprise bean, deben emplearse DJAR distintos para contener enterprise beans diferenciados para las aplicaciones empresariales en el ámbito.

Migración de componentes de enterprise bean

CICSPlex SM brinda un conjunto de herramientas para ayudar a los usuarios a migrar sus definiciones de RDO (definición de recursos en línea) desde el CSD de CICS al repositorio de datos de CICSPlex SM.

Este conjunto de herramientas consta de un programa de salida para el programa de utilidad del CSD fuera de línea de CICS y algún JCL de ejemplo para ejecutarlo; consulte Extracting records from the CSD, en el manual *Aplicaciones de negocio de gestión de CICSPlex System Manager*.

Esta salida de CICSPlex SM reconocerá definiciones CORBASERVER y DJAR en un CSD y generará las sentencias BAS CREATE EJCODEF y CREATE EJDJDEF apropiadas para la entrada mediante el proceso BatchRep de CICSPlex SM. Todas las reglas de selección normales para identificación de recursos se pueden aplicar a estos tipos de recurso de EJB.

Soporte de consultas de CICSPlex SM para enterprise beans

CICSPlex SM puede gestionar las instancias instaladas de CorbaServer y de DJAR mediante cualquiera de las tres interfaces descritas en "Soporte de CICSPlex SM para enterprise beans" en la página 376. Todos los servicios de operador interactivos que se proporcionan mediante las transacciones CEMT y CEOT de CICS tienen una réplica funcional en CICSPlex SM mediante la interfaz de usuario web (WUI). En cualquier caso, los objetos de CICS instalados correlacionados por CICSPlex SM son:

- EJCOSE: instancias de CorbaServer.
- EJDJAR: instancias de archivo JAR desplegado por CICS.

Además, mediante estos objetos puede listarse cualquier enterprise bean ejecutable:

- EJCOBEAN: Enterprise JavaBeans asociados directamente con un CorbaServer.
- EJDJBEAN: enterprise beans asociados directamente con un DJAR.

Estos dos objetos describen una estructura de enterprise bean: uno se especifica mediante un nombre de CorbaServer y el otro mediante un identificador de DJAR. En ambos casos, el único contenido de enterprise bean disponible para consulta es el nombre del CorbaServer, el nombre del DJAR y el nombre del enterprise bean de hasta 240 caracteres de longitud. La especificación de Enterprise JavaBean indica que los nombres de enterprise bean pueden ser mucho más largos, pero la implementación de CICS los limita a 240 bytes. Un detalle adicional que las consultas de CICSPlex SM brindan y que no brinda una consulta de CICS estándar es un recuento de los beans disponibles en cualquier DJAR o CorbaServer concreto. Cuando se despliega un nuevo conjunto de enterprise beans mediante un DJAR en un CorbaServer concreto, el recuento de enterprise beans puede brindar una confirmación instantánea sobre la disponibilidad de los enterprise beans en cuestión. El valor aumenta en función del número de enterprise beans que se aceptan mediante el proceso de instalación del DJAR.

Otros objetos de CICS asociados con Enterprise Java que se pueden consultar mediante CPSM son:

- TCPIPS: instancias de TCPIPSERVICE.
- RQMODEL: instancias de REQUESTMODEL.
- LOCTRAN: instancias de transacción local.
- UOWORK: instancias de unidad de trabajo.

- UOWLINK: instancias de enlace de unidad de trabajo (UOWLINK).
- PROGRAM: instancias de programa.

Todos estos objetos incluyen atributos que son relevantes para la gestión y la ejecución de enterprise beans.

Tipos de consulta disponibles para objetos de enterprise bean

Hay varias maneras de consultar el estado de los objetos EJB con CICSPlex SM.

La interfaz de programación de aplicaciones de CICSPlex SM

Para consultar los objetos EJB utilizando los mandatos disponibles de la API de CICSPlex SM, consulte la *Referencia de programación de la aplicación de CICSPlex System Manager*. Consulte también los detalles de los atributos y de las acciones que se permiten en cada objeto de CICSPlex SM en la *Referencia de tablas de recursos de CICSPlex System Manager Vol 1*.

La interfaz de usuario web de CICSPlex SM

Para consultar los objetos EJB utilizando la WUI, consulte la *Guía de la interfaz de usuario web de CICSPlex System Manager*.

La interfaz de usuario web tiene un conjunto inicial que consta de un conjunto de menús y de paneles. Este conjunto inicial incluye un conjunto de vistas de componentes de Enterprise Java.

Utilización de CICSPlex SM para gestionar cargas de trabajo de EJB

Una de las funciones estándar del componente CICSPlex SM es el recurso para equilibrar y separar transacciones de CICS en un entorno MRO, lo que se conoce como gestión de carga de trabajo (WLM).

Este recurso resulta de lo más adecuado para la gestión de cargas de trabajo de EJB, en las que los enterprise beans se ejecutan en un entorno de CorbaServer distribuido o lógico. En su configuración más simple, CICSPlex SM puede equilibrar la carga de trabajo de ejecución de un enterprise bean en una serie de regiones propietarias de la aplicación (AOR), en función de objetivos de rendimiento y de algoritmos de estabilidad establecidos por las definiciones del usuario. Estas funciones se implementan cuando el programa de salida de direccionamiento distribuido proporcionado por CICSPlex SM (EYU9XLOP) se indica como el parámetro DSRTPGM en los parámetros de inicialización del sistema de las regiones de escucha y las AOR participantes (consulte Balancing an enterprise bean workload, en el manual *Cargas de trabajo de gestión de CICSPlex System Manager*).

Loa algoritmos utilizados por CICSPlex SM para seleccionar AOR adecuadas para la ejecución de enterprise beans se ha establecido y ajustado desde el nacimiento del producto. Sin embargo, los usuarios pueden optar por desarrollar su propio programa de algoritmos de direccionamiento y sustituir la versión de CICSPlex SM (EYU9WRAM) que se proporciona, si tienen que hacerlo.

Direccionamiento de la carga de trabajo:

El direccionamiento de la carga de trabajo de CICSPlex SM brinda una función que selecciona la AOR más adecuada para albergar la ejecución de un enterprise bean, según criterios de selección predeterminados.

El proceso de selección de AOR evalúa toda la actividad de ejecución simultánea, en las regiones designadas como posibles destinos del direccionamiento, y selecciona la región más adecuada en términos de carga de trabajo de ejecución y

de estabilidad de región en el punto de consulta. Esto *no* es lo mismo que la selección cíclica de una AOR de entre todas las disponibles en un ámbito disponible para beans ejecutados en serie. *Es* la evaluación de todas las transacciones activas en el ámbito del WLM en el momento en el que una nueva transacción (enterprise bean) está a punto de ejecutarse y la selección de la región menos cargada o más estable para albergar la ejecución del objeto. La implementación de direccionamiento de carga de trabajo básico para toda la producción de Enterprise Java tiene los siguientes requisitos previos:

- Las definiciones de TCP/IP necesarias están instaladas en las regiones de escucha designadas.
- DSRTPGM=EYU9XLOP está especificado como parámetro de SIT en todas las regiones de escucha y AOR.
- MASPLTWAIT(YES) está incluido como EYUPARM en todas las regiones de escucha.
- La transacción del procesador de solicitudes (la transacción predeterminada es CIRP) se ha definido de forma dinámica para las regiones de escucha y de forma estática para las AOR.
- Las definiciones necesarias de CorbaServer y DJAR están instaladas (bien mediante BAS o bien mediante CEDA) para establecer el entorno de EJB ejecutable.
- Los enterprise beans se han desplegado y están en servicio.

Cuando se hayan cumplido estos criterios, defina un objeto de especificación de carga de trabajo (WLMSPEC) y especifique las AOR como el ámbito de destino. Luego puede instalar el objeto WLMSPEC en todas las regiones de escucha y AOR que se vayan a unir a la carga de trabajo. Cuando se haya instalado el WLMSPEC, todas las regiones que englobe tendrán sus cargas de trabajo de EJB direccionadas, tras haber sido reiniciadas. Se proporciona un ejemplo detallado del direccionamiento de carga de trabajo de enterprise bean en Balancing an enterprise bean workload, en el manual *Cargas de trabajo de gestión de CICSPlex System Manager*.

Separación de carga de trabajo:

La separación de carga de trabajo es la función de la gestión de carga de trabajo (WLM) que hace que las transacciones que cumplen criterios de selección designados previamente se dirijan a ámbitos de destino específicos.

El ámbito de destino de un elemento de carga de trabajo separada puede variar de una única región propietaria de la aplicación (AOR) a un gran grupo de AOR que conste de varias regiones de CICS. Si un grupo de AOR es el destino, el algoritmo de direccionamiento se aplicará a la región más adecuada de las definidas para ello. Para implementar una carga de trabajo que incluya enterprise beans separados, primero es necesario establecer el requisito previo de direccionamiento de carga de trabajo que se describe en “Direccionamiento de la carga de trabajo” en la página 379. Esa configuración tiene que aumentarse mediante los siguientes componentes adicionales:

- Una transacción CIRP clonada para cada enterprise bean que se tenga que separar (un copia simple de la definición existente con un nuevo nombre).
- Un modelo de solicitud para cada enterprise bean que separar, para asociarlo con una de las transacciones CIRP clonadas.

Esto permite que se establezcan los entornos de CICS y EJB, lo que habilita la separación de enterprise beans. Luego deberán crearse las definiciones del WLM

para implementarlo. Esto conlleva identificar las transacciones CIRP clonadas como objetos de interés y asociarlas con los ámbitos de destino necesarios mediante una serie de definiciones de WLM. Estas definiciones de WLM se beben asociar con una especificación WLM general, mediante un grupo de WLM intermedio, y luego dicha especificación se debe añadir al grupo de CICS que incluya todas las regiones de escucha y las AOR que van a participar en la carga de trabajo. Se incluye un ejemplo detallado de separación de carga de trabajo de enterprise bean en Separating enterprise beans in a workload, en el manual *Cargas de trabajo de gestión de CICSPlex System Manager*.

Supervisor de recursos de CICSPlex SM para enterprise beans

La supervisión de CICSPlex SM permite la recopilación de datos relacionados con el rendimiento, en intervalos definidos por el usuario, para instancias de recurso indicadas en un conjunto de sistemas CICS.

Actualmente no se registra ningún dato relacionado con el rendimiento para objetos EJB específicos (CorbaServer y DJAR). Sin embargo, los datos de rendimiento para las transacciones del receptor de peticiones IIOP y del procesador de solicitudes están disponible de la forma normal, por lo que el rendimiento de ejecución de enterprise beans se puede supervisar mediante un código de transacción asociado (consulte la *Referencia de vistas de supervisión de CICSPlex System Manager*). Los usuarios necesitarán modelos de solicitud y clones de CIRP para cada bean que haya que supervisar, del mismo modo que para la separación de carga de trabajo de enterprise beans, descrita en "Separación de carga de trabajo" en la página 380. Sin embargo, la supervisión de CICSPlex SM no está integrada en el ámbito lógico de BAS, por lo que el ámbito de las vistas de supervisión debe definirse al grupo de CICS físico que abarque las regiones que supervisar, en lugar de a la descripción de recursos BAS que instalara las definiciones de transacción. En Collecting statistics using CICSPlex SM monitoring, en el manual *Conceptos y planificación de CICSPlex System Manager*, se proporciona una visión general de la función de supervisión. En Preparing to monitor resources, en el manual *Uso de recursos de gestión de CICSPlex System Manager*, se brindan detalles completos de la función de supervisión.

Consideraciones sobre el análisis en tiempo real de CICSPlex SM para enterprise beans

La función de análisis en tiempo real (RTA) de CICSPlex SM brinda notificación automática y externa de las condiciones por las que los usuario hayan expresado su interés.

El análisis en tiempo real se puede dividir en diversos subcomponentes:

- Supervisión de disponibilidad del sistema (SAM): supervisa las regiones CICS durante sus horas de disponibilidad planeadas y genera notificaciones cuando no se reciben respuestas de una región que se espera que esté activa.
- Supervisión de recursos MAS (MRM): supervisa el estado de cualquier recurso de CICS que se pueda consultar y genera notificaciones cuando ese estado varía de una norma predeterminada.
- Supervisión de punto de análisis (APM): replica la función de MRM, excepto que analiza los estados en un nivel de CICSplex, en lugar de en una región CICS específica. La APM resulta especialmente útil en entornos que utilizan AOR clonadas, en los que las regiones son idénticas y en los que una sola notificación es suficiente para alertarle si se produce un problema general.

Evidentemente, la SAM es una función útil para informar de la disponibilidad de regiones CICS, independientemente de si están designadas como regiones de

escucha o como AOR. Si ejecuta enterprise beans en un entorno distribuido, la MRM puede resultarle más útil para la supervisión del estado de CorbaServer y DJAR, en lugar de las funciones basadas en regiones de la APM. Sin embargo, tenga en cuenta que no puede supervisar los propios objetos de enterprise bean (EJCOBEAN y EJDJBEAN) en RTA. Las consultas de enterprise bean pueden especificarse solo en sus nombres de CorbaServer o DJAR correspondientes. No se pueden realizar consultas específicas sobre el nombre del enterprise bean. En Exception reporting using real-time analysis (RTA), en el manual *Conceptos y planificación de CICSPlex System Manager*, se proporciona una visión general de la función RTA. En Preparing to perform real-time analysis, también en el manual *Uso de recursos de gestión de CICSPlex System Manager*, se proporcionan detalles completos de la función RTA.

CICS y el protocolo Inter-ORB de Internet (IIOP)

Esta sección indica lo que necesita saber para configurar CICS para el soporte de aplicaciones IIOP distribuidas.

Soporte de IIOP en CICS

El protocolo Inter-ORB de Internet (IIOP) es una implementación basada en TCP/IP del protocolo Inter-ORB general (GIOP) que define formatos y protocolos para aplicaciones distribuidas.

Forma parte de la Common Object Request Broker Architecture (CORBA). Tanto el sistema cliente como el sistema de servidor necesitan un intermediario para solicitudes de objetos (ORB) de CORBA para implementar la interoperatividad de IIOP.

La Common Object Request Broker Architecture es una especificación para una arquitectura estándar orientada a objetos para aplicaciones distribuidas. Fue definido por un consorcio de más de 500 organizaciones de tecnologías de la información llamado Grupo de gestión de objetos (OMG). Puede leer el documento de la *Arquitectura y especificación* de CORBA en su sitio web: http://www.omg.org/

CICS proporciona un ORB y soporte para IIOP definido por CORBA 2.3.

El intermediario para solicitudes de objetos (ORB)

CORBA utiliza un **intermediario** para gestionar las solicitudes entre clientes y servidores en el sistema. El intermediario elige el mejor servidor para cumplir la solicitud del cliente y separa la **interfaz** que ve el cliente de la **implementación** del servidor.

El intermediario, conocido como ORB, intercepta las llamadas a método del cliente y es responsable de buscar objetos que puedan implementar solicitudes, pasarles parámetros, invocar sus métodos y devolver resultados. El cliente tiene que saber dónde está ubicado el objeto, su lenguaje de programación, su sistema operativo o cualquier otro aspecto del sistema que no forme parte de la interfaz del objeto.

De este modo, el ORB proporciona interoperatividad entre aplicaciones que están en distintas máquinas y en entornos distribuidos variados e interconecta varios sistemas de objetos.

El ORB de CICS implementa el siguiente nivel de función:

- Soporte para CORBA versión 2.3, *excepto para*:
  - Objetos CORBA con estado (solo se soportan objetos CORBA sin estado).

    Nota: La única excepción a esta regla son los beans de sesión con estado, que *sí están* soportados.
  - La interfaz de invocación dinámica (DII).
  - La interfaz de esqueletos dinámica (DSI).
  - Fragmentos de GIOP 1.1.
  - El adaptador de objetos portátil (POA).
  - GIOP bidireccional
- Soporte para IIOP 1.2, incluidos fragmentos de GIOP 1.2.
- Soporte para solicitudes de IIOP de entrada y salientes. Las aplicaciones de IIOP pueden funcionar como cliente y como servidor.
- Soporte para **objetos transaccionales**. Las invocaciones de método de CICS pueden participar en transacciones distribuidas del servicio de transacción de objetos (OTS). Si un cliente llama a una aplicación IIOP dentro del ámbito de una transacción del OTS, la información sobre dicha transacción fluye como un parámetro extra en la llamada a IIOP. Si el ORB del cliente envía un contexto de servicio de transacción del OTS y el objeto de destino CORBA sin estado implementa `CosTransactions::TransactionalObject`, dicho objeto se trata como transaccional.

  Nota: Una **transacción del OTS** es una unidad de trabajo distribuida y no una instancia de transacciones CICS ni una definición de recurso. Para obtener una descripción de una transacción de CICS, consulte "Transacciones de CICS" en la página 30.

La función del ORB se implementa en CICS mediante:

- La región de escucha del dominio de sockets de CICS.
- El receptor de peticiones de IIOP de CICS.
- El procesador de solicitudes de IIOP de CICS.

Modelos de aplicación IIOP de CICS

Las aplicaciones IIOP son programas orientado a objetos cliente/servidor que se ejecutan en una red TCP/IP.

CICS soporta los siguientes tipos de aplicación IIOP:

Objetos CORBA sin estado
: Los objetos CORBA sin estado son aplicaciones de servidor Java que se comunican con una aplicación cliente utilizando el protocolo IIOP. No se mantiene ningún estado en los atributos del objeto entre invocaciones sucesivas de métodos; el estado se inicializa al principio de cada llamada a método y es referenciado mediante parámetros explícitos.

  Los objetos CORBA sin estado pueden recibir solicitudes de entrada de un cliente y también realizar solicitudes IIOP salientes.

  Los objetos CORBA sin estado de CICS se ejecutan en una JVM de CICS.

  Puede leer más sobre los objetos CORBA sin estado de CICS en "Objetos CORBA sin estado" en la página 211.

Enterprise beans
: Los enterprise beans son aplicaciones de servidor Java portátiles que utilizan

las interfaces definidas por la *Enterprise JavaBeans Specification, versión 1.1*. CICS ha implementado estas interfaces correlacionándolas con los servicios de CICS subyacentes.

Los enterprise beans se comunican mediante la interfaz de invocación a método remoto (RMI) de Java. CICS soporta RMI sobre IIOP, mediante un intermediario para solicitudes de objetos (ORB) de CORBA.

Los enterprise beans se pueden enlazar con otros programas de CICS utilizando el **CCI Connector for CICS TS**. También puede desarrollar enterprise beans que utilicen la biblioteca de clases de JCICS para acceder a servicios o programas de CICS directamente, pero estas aplicaciones de servidor no se pueden transportar a una plataforma que no sea CICS.

Los enterprise beans se ejecutan en una JVM en agrupación.

Puede leer más sobre los enterprise beans en "¿Qué son los enterprise beans?" en la página 232.

Terminología común de CORBA

Estos términos se utilizan en este segmento de información.

CORBA
La Common Object Request Broker Architecture (arquitectura común de intermediario para solicitudes de objetos). Una arquitectura y una especificación para computación distribuida orientada a objetos.

GIOP El protocolo Inter-ORB general. La especificación de representación de datos y el protocolo de interoperatividad de CORBA. Define cómo se comunican distintos ORB; no define qué protocolo de transporte hay que utilizar.

IDL Interface Definition Language (idioma de definición de interfaz). Un idioma de definición que se utiliza en CORBA para describir las características y el comportamiento de un tipo de objeto, incluidas las operaciones que se pueden realizar en él.

IIOP El protocolo Inter-ORB de Internet. Define cómo enviar mensajes GIOP sobre una capa de transporte TCP/IP. IIOP es GIOP sobre TCP/IP.

Interfaz
Describe las características y el comportamiento de un tipo de objeto, incluidas las operaciones que se pueden realizar en dichos objetos. Se correlaciona con una **clase** Java. En terminología CORBA, la solicitud del cliente especifica, en IDL, una interfaz que define el objeto de servidor.

IOR Interoperable Object Reference (referencia de objeto interoperable). Una referencia "mediante series" a un objeto CORBA remoto. Lo publica el ORB del servidor. La aplicación cliente debe tener acceso a la IOR en tiempo de ejecución. El ORB del cliente puede deconstruir el IOR para determinar (entre otras cosas) la ubicación del ORB remoto y del objeto, la versión máxima de GIOP soportada por el ORB remoto y cualquier servicio de CORBA relevante que esté soportado por el ORB remoto.

Módulo
Un conjunto de empaquetado IDL que contiene interfaces. Se correlaciona con un **paquete** Java.

OMG El grupo de gestión de objetos. El conjunto de organizaciones de software que ha definido la arquitectura CORBA.

Operación
Una acción que se puede realizar en un objeto. Se correlaciona con un

método Java. En terminología CORBA, el cliente solicita una operación, definida en IDL, que se correlaciona con un método en el objeto de servidor.

ORB El intermediario para solicitudes de objetos. Un componente del sistema CORBA que actúa como intermediario entre la aplicaciones de cliente y de servidor. Tanto la plataforma de cliente como la de servidor necesitan un ORB; cada uno se adapta a un entorno específico, pero soporta protocolos CORBA e IDL comunes.

RMI-IIOP
La especificación y el protocolo de invocación a método remoto (RMI) sobre IIOP. La especificación define cómo hacer que la arquitectura de aplicación RMI específica de Java interopere utilizando protocolos CORBA. Se trata del protocolo de comunicación que utilizan los enterprise beans.

Esqueleto
Un fragmento de código generado por el compilador IDL del servidor. Lo utiliza el ORB del servidor para analizar un mensaje de una llamada a método de un objeto local (para el servidor).

Apéndice o proxy
Un fragmento de código generado por el compilador IDL o RMI del cliente. Lo utiliza la aplicación cliente para invocar métodos en el objeto remoto. La clase de apéndice (stub) llama a métodos en el ORB del cliente, que a su vez envía solicitudes de método remoto al ORB del servidor. La clase de apéndice se debe generar para el ORB específico del cliente con el que se va a utilizar. Si utiliza ORB de cliente de distintos proveedores, debe asegurarse de utilizar apéndices del lado de cliente generados mediante la herramientas incluidas con el ORB de cliente correcto.

Enlace Un fragmento de código generado por el compilador RMI. Lo utiliza el ORB del servidor para analizar un mensaje de una llamada a método de un objeto local (para el servidor).

El flujo de solicitudes de IIOP

Este diagrama muestra el flujo de ejecución de una solicitud entrante.

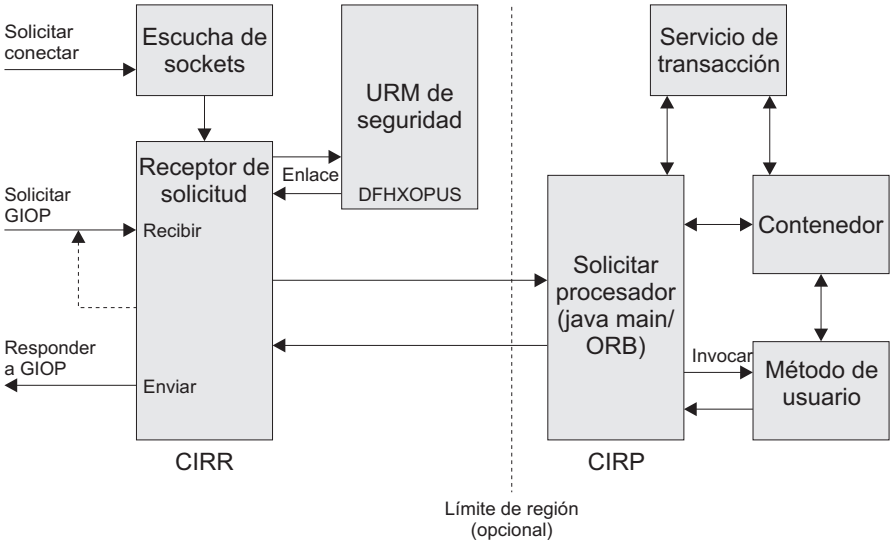


Figura 34. Flujo de ejecución de solicitudes de IIOP

La región de escucha de TCP/IP

La región de escucha de TCP/IP de CICS supervisa los puertos especificados para solicitudes de entrada. Se especifican los puertos de IIOP y se configura la región de escucha definiendo e instalando recursos de TCPIPSERVICE.

La región de escucha recibe la solicitud entrante e inicia la transacción especificada en la definición de TCPIPSERVICE para ese puerto. En el caso de servicios de IIOP, esta definición de recurso de transacción debe tener el atributo de programa definido como DFHIIRRS, el programa del receptor de peticiones. El nombre predeterminado de la transacción es CIRR.

Receptor de peticiones

El receptor de peticiones recupera la solicitud entrante y examina el contenido de la secuencia de mensajes con formato de GIOP. Se pueden recibir los siguientes tipos de mensaje de GIOP, y se gestionan de la siguiente forma:

Solicitud

- Se determina un ID de usuario de CICS a partir de los parámetros de la capa de sockets seguros (SSL) o llamando a un programa sustituible por el usuario de CICS especificado por la definición de recurso TCPIPSERVICE. El ID de usuario de CICS se utiliza para la autorización de la solicitud por parte del procesador de solicitudes.
- El TRANSID de CICS se determina a partir del contenido del mensaje mediante la comparación con las definiciones de recurso REQUESTMODEL instaladas. El TRANSID de CICS define parámetros de ejecución que se utilizan si se crea una nueva instancia del procesador de solicitudes para gestionar la solicitud.
- La solicitud se pasa al procesador de solicitudes utilizando una **secuencia de solicitudes**, que es un mecanismo de direccionamiento interno de CICS. La clave de objeto de la solicitud, o cualquier contexto de servicio de transacción, determina si la solicitud se va a enviar a un procesador existente.

Nota: Una *transacción* en este contexto significa una unidad de trabajo y se gestiona utilizando la especificación del **servicio de transacción de objetos** (OTS).

La lógica de gestión de solicitudes utiliza un directorio para determinar si una solicitud de IIOP debe dirigirse a una instancia de procesador existente (mediante su secuencia de solicitudes asociada). El directorio, DFHEJDIR, relaciona secuencias de solicitudes (e instancias del procesador de solicitudes) con transacciones del OTS y con las claves de objeto de beans de sesión con estado que gestionan sus propias transacciones. DFHEJDIR es un archivo de CICS recuperable.

- Los fragmentos entrantes de GIOP 1.1 se rechazan con un mensaje MessageError de GIOP.

`LocateRequest`
Localice solicitudes que no tienen **operación** ni parámetro. Se pasan a una nueva instancia del procesador de solicitudes.

`CancelRequest`
Una solicitud de cancelación avisa a un servidor de que el cliente ya no espera una respuesta a una solicitud pendiente o a un mensaje de LocateRequest. Este es solo un mensaje de aviso, no se espera respuesta. Una solicitud de cancelación recibida durante el proceso de fragmento hace que la solicitud en curso se termine. Todas las demás solicitudes de cancelación se ignoran.

`MessageError`
Un error de mensaje indica que el cliente no ha reconocido una respuesta que le ha enviado el receptor de peticiones. Este error se registra para fines de diagnóstico y se envía un mensaje CloseConnection a la conexión.

`Fragmentos`
Un fragmento es una continuación de una solicitud o de una respuesta. Contiene una cabecera de mensaje de GIOP seguida por datos. Los fragmentos entrantes de GIOP 1.1 se rechazan con un mensaje MessageError de GIOP.

El enlace desde el receptor de peticiones puede aprovechar los servicios de direccionamiento dinámico de CICS para proporcionar equilibrio de carga en el CICSplex.

El receptor de peticiones CIRR termina cuando no tiene más trabajo que hacer. (Es decir, CIRR termina cuando no quedan solicitudes pendientes de GIOP que leer del TCPIPSERVICE y no quedan respuestas pendientes que enviar desde solicitudes anteriores. Si llega nueva carga de trabajo para el TCPIPSERVICE tras terminar la tarea de CIRR, se inicia una nueva tarea de CIRR).

`Procesador de solicitudes`
El procesador de solicitudes gestiona la ejecución de la solicitud de IIOP. Este:

- Localiza el objeto identificado por la solicitud.
- Para una solicitud de enterprise bean, llama al contenedor para que procese el método de bean.
- Para una solicitud sobre un objeto CORBA sin estado, procesa la propia solicitud (aunque el servicio de transacción puede estar implicado también).

La instancia del procesador de solicitudes que gestiona cada solicitud de IIOP la configura una definición de recurso CORBASERVER.

IIOP en un sysplex

Puede implementar un servidor CORBA de CICS en una única región CICS. Sin embargo, en un sysplex es probable que desee crear un servidor que conste de varias regiones.

El uso de varias regiones hace que el error de una única región no sea tan crítico y permite utilizar el direccionamiento de carga de trabajo. Un servidor lógico de CICS consta de una o más regiones CICS configuradas para comportarse como un único servidor.

Por lo general, un servidor lógico de CICS consta de los siguientes componentes:

- Un conjunto de regiones de escucha clonadas definidas por definiciones TCPIPSERVICE idénticas para escuchar las solicitudes de IIOP entrantes.
- Un conjunto de regiones propietarias de la aplicación (AOR) clonadas, cada una de las cuales soporta un conjunto idéntico de aplicaciones de IIOP o de clases de enterprise bean en un servidor CORBA definido de forma idéntica. Varios métodos para la misma transacción del OTS (servicio de transacciones de objetos) están dirigidos a la misma AOR. Cada AOR debe tener definiciones TCPIPSERVICE que coincidan en las regiones de escucha correspondientes.

Las regiones de escucha y las AOR pueden separarse o combinarse en AOR de escucha.Debe especificar los siguientes parámetros de inicialización del sistema:

IIOPLISTENER=YES
: Especifique este valor en una región de escucha, o en una AOR de escucha combinada. YES es el valor predeterminado.

IIOPLISTENER=NO
: Especifique este valor en una AOR que no sea también una región de escucha.

Direccionamiento de carga de trabajo de solicitudes IIOP

Para direccionar las conexiones de cliente en regiones de escucha, puede utilizar direccionamiento de IP u optimización de conexiones utilizando el registro del Sistema de nombres de dominio (DNS): Para direccionar las transacciones del Servicio de transacción de objetos (OTS) en un conjunto de regiones propietarias de la aplicación clonado (AORs), utilice el direccionamiento distribuido. Para implementar el direccionamiento distribuido, puede utilizar CICSPlex System Manager o una versión personalizada del programa de direccionamiento distribuido de CICS, DFHDSRP.

Optimización de la conexión mediante Sistema de nombres de dominio (DNS)
: La optimización de la conexión es una técnica que utiliza DNS para equilibrar las conexiones IP en un dominio de sysplex. Con DNS, se inician varios sistemas CICS para que escuchen solicitudes de IIOP en el mismo puerto (utilizando direcciones IP virtuales), y se registran en el gestor de carga de trabajo (WLM) de MVS. Cada solicitud IIOP de cliente contiene un nombre de host genérico y un número de puerto. Este nombre de host se resuelve en una dirección IP mediante los servicios de DNS y WLM.

  La optimización de la conexión utilizando WLM se describe en *z/OS Communications Server: guía de configuración de IP*.

Direccionamiento distribuido
: El direccionamiento distribuido se utiliza para direccionar llamadas a método para enterprise beans y objetos CORBA sin estado en un conjunto de regiones propietarias de la aplicación de CICS. La selección dinámica del destino la realiza el gestor de cargas de trabajo (CICSPlex SM o un programa de

direccionamiento distribuido escrito por el usuario) que selecciona la región de aplicación menos cargada o más eficiente. CICS invoca el gestor de carga de trabajo para solicitudes de métodos que se ejecutarán en una transacción del OTS nueva (o no), pero no para solicitudes de métodos que se ejecutarán en una transacción del OTS ya existente; estas las dirige automáticamente a la AOR en la que se ejecuta la transacción del OTS existente. Consulte Writing a distributed routing program , en la *Guía de personalización de CICS*, para obtener orientación sobre la escritura de un programa de direccionamiento distribuido personalizado. Consulte Workload management and dynamic routing, en el manual *Cargas de trabajo de gestión de CICSPlex System Manager*, para obtener información sobre gestión de carga de trabajo de CICSPlex SM.

El siguiente diagrama muestra un servidor lógico de CICS. En este ejemplo, las regiones de escucha y las AOR están en grupos distintos, se utiliza optimización de la conexión para equilibrar las conexiones de cliente en las regiones de escucha y se emplea direccionamiento distribuido para direccionar las transacciones del OTS en las AOR.

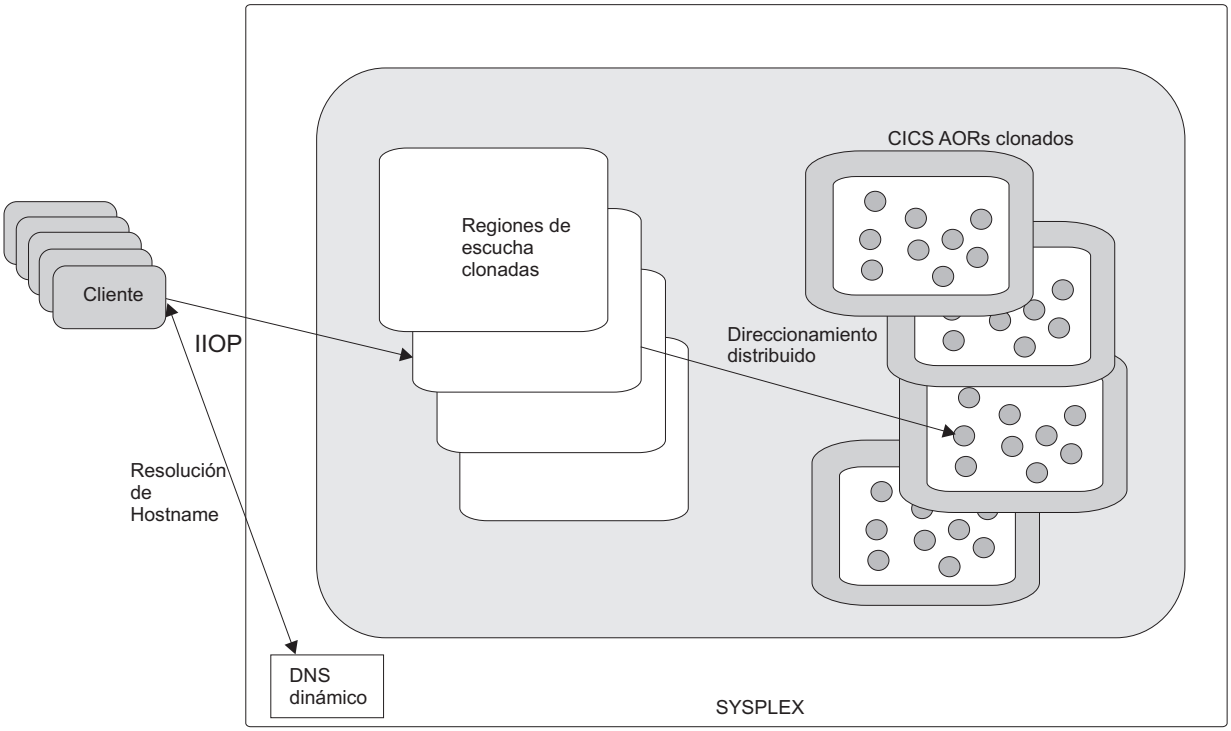


Figura 35. Un servidor lógico de CICS.. En este ejemplo, el servidor lógico consta de un conjunto de regiones de "escucha" clonadas y de un conjunto de AOR clonadas. La optimización de la conexión mediante registro de DNS dinámico se utiliza para direccionar las conexiones de cliente a las regiones de escucha. El direccionamiento distribuido se utiliza para equilibrar transacciones del OTS en las AOR.

Optimización de la conexión mediante Sistema de nombres de dominio (DNS)

La optimización de la conexión es una técnica que utiliza DNS para equilibrar las conexiones IP y la carga de trabajo en un dominio de sysplex.

En términos del DNS, un sysplex es un subdomino que se añade al espacio de nombres del DNS. La optimización de la conexión amplía el concepto de un "nombre de host de DNS" a clústeres o a grupos de hosts o de aplicaciones de servidor. Las aplicaciones de servidor dentro del mismo grupo se considera que

proporcionan un servicio equivalente. La optimización de la conexión utiliza ordenación basada en la carga para determinar qué direcciones devolver para un clúster determinado .

Registro de optimización de conexión:

Las aplicaciones de servidor se registran en el gestor de carga de trabajo (WLM) de MVS, que cuantifica la disponibilidad de recursos de servidor en un sysplex.

El WLM se debe configurar en modalidad de objetivo en todos los hosts del sysplex. Las pilas TCP/IP también se pueden registrar en el WLM para brindar información sobre las direcciones IP iniciadas, o se pueden utilizar definiciones estáticas si las pilas no soportan registro. Al registrarse, las aplicaciones de servidor proporcionan la siguiente información:

Nombre de grupo
Es el nombre de un clúster de aplicaciones de servidor equivalentes en un sysplex. Es el nombre dentro del dominio de sysplex que las aplicaciones cliente utilizan para acceder a aplicaciones de servidor. CICS utiliza el parámetro DNSGROUP de la definición de recurso TCPIPSERVICE como el nombre de grupo que registrar en el WLM.

Nombre de servidor
Se trata del nombre de la instancia de aplicación de servidor. El nombre de servidor debe ser exclusivo entre todos los servidores que comparten el mismo nombre de grupo. Una instancia de aplicación de servidor puede pertenecer a más de un grupo. CICS registra en el WLM utilizando el ID_aplic genérico de la región como lo especifica el parámetro de inicialización del sistema APPLID.

Nombre de host
Se trata del nombre de host de la pila TCP/IP en la que se ejecutar la aplicación de servidor. Durante el inicio, CICS llama a la función TCP/IP *gethostbyaddr* para determinar el nombre de host de la máquina en la que se ejecuta y lo pasa al WLM para registro.

Ejemplo de resolución de nombres:

Este ejemplo muestra un CICSplex que consta de cuatro regiones CICS, cada una de las cuales se ejecuta en una máquina distinta de un sysplex.

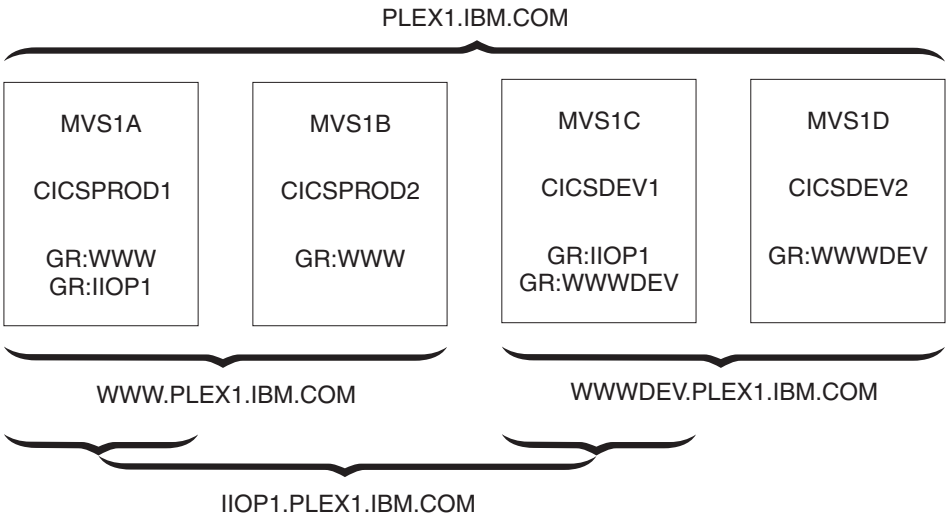


Figura 36. CICSplex que utiliza optimización de la conexión DNS

Los sistemas de MVS se llaman MVS1A, MVS1B, MVS1C y MVS1D,, y las regiones CICS tienen los ID de aplicación CICSPROD1, CICSPROD2, CICSDEV1 y CICSDEV2.

El sysplex se define para el DNS para tener el nombre PLEX1, y cada máquina de MVS tiene una dirección IP única. El diagrama describe los nombres que una máquina cliente podría utilizar para acceder a las regiones CICS en base a las siguientes definiciones de recurso instaladas en cada CICS.

- La región CICSPROD1 que se ejecuta en la máquina MVS1A tiene dos recursos TCPIPSERVICE, uno que especifica un group_name (nombre de grupo) de WWW y el segundo que especifica un group_name de IIOP1.
- La región CICSPROD2 que se ejecuta en la máquina MVS1B tiene un recurso TCPIPSERVICE, que especifica un group_name de WWW.
- La región CICSDEV1 que se ejecuta e la máquina MVS1C tiene dos recursos TCPIPSERVICE, uno que especifica un group_name de IIOP1 y el segundo que especifica un group_name de WWWDEV.
- La región CICSDEV2 que se ejecuta en la máquina MVS1D tiene un recurso TCPIPSERVICE, que especifica un group_name de WWWDEV.

El cliente puede acceder a los siguientes nombres:

- PLEX1.IBM.COM devuelve la dirección IP de cualquier máquina que haya en el sysplex.
- WWW.PLEX1.IBM.COM devuelve la dirección de MVS1A o de MVS1B.
- IIOP1.PLEX1.IBM.COM devuelve la dirección de MVS1A o de MVS1C.
- WWWDEV.PLEX1.IBM.COM devuelve la dirección de MVS1C o de MVS1D.

También puede dirigirse a regiones CICS individuales dentro de un grupo mediante sus ID de aplicación (o sus nombres de servidor). Por ejemplo, CICSPROD1.WWW.PLEX1.IBM.COM devuelve la dirección de MVS1A. Esta dirección es el equivalente a MVS1A.PLEX1.IBM.COM, pero el cliente no tiene que saber en qué máquina se está ejecutando el servidor CICSPROD1, solo que CICSPROD1 forma parte del grupo WWW.

Como estos nombres pasan a estar disponibles de forma dinámica a medida que las regiones CICS se registran en el WLM, añadir más regiones CICS y más

máquinas de MVS no da como resultado más tareas de administración. Utilizar nombres de host genéricos (como WWWDEV.PLEX1.IBM.COM) separa a aplicaciones cliente de regiones CICS y hosts de MVS específicos, lo que mejora la disponibilidad y la escalabilidad.

Definición de recurso para optimización de la conexión DNS:

Es necesario definir estas opciones de TCPIPSERVICE para los puertos TCP/IP que utilizan optimización de la conexión DNS.

DNSGROUP
Especifica el parámetro de ubicación que se pasa en la llamada de registro IWMSRSRG al gestor de carga de trabajo. El valor puede tener hasta 18 caracteres de longitud y se ignoran los de espacios en blanco finales.

Este parámetro se denomina `nombre_grupo` en la documentación sobre DNS de TCP/IP. Es el nombre genérico de un clúster de aplicaciones de servidor equivalentes en un sysplex. También es el nombre dentro del dominio del sysplex que los clientes utilizan para acceder a CICS TCPIPSERVICE.

Se permite que más de un recurso TCPIPSERVICE especifique el mismo nombre de grupo.

La llamada de registro se efectúa a WLM cuando se abre el primer servicio que tiene especificado este nombre de grupo. Los servicios subsiguientes con el mismo nombre de grupo no hacen que se realicen más llamadas de registro.

La acción de anular el registro la dicta el atributo GRPCRITICAL, como se describe a continuación. También es posible anular el registro de CICS de manera explícita desde un grupo emitiendo el mandato **SET TCPIPSERVICE DNSSTATUS DEREGISTERED** del terminal maestro (CEMT) o de **EXEC CICS**, o utilizando el mandato equivalente de CICSPlex SM.

GRPCRITICAL
Marca el servicio como un miembro crítico del grupo de DNS, de forma que el cierre o la anomalía de este servicio provoca que se realice una llamada de anulación de registro a WLM para este nombre de grupo.

El valor predeterminado es NO, lo que permite que dos o más servicios del mismo grupo sufran una anomalía de forma independiente y aún así CICS siga registrado en el grupo. Solo cuando se ha cerrado el último servicio de un grupo se realiza la llamada de anulación de registro al WLM, si no se ha hecho ya explícitamente.

Varios servicios con el mismo nombre de grupo pueden tener distintos valores de grpcritical. Los servicios que especifican GRPCRITICAL(NO) se pueden cerrar ni sufrir un error sin provocar una anulación de registro. Si un servicio con GRPCRITICAL(YES) se cierra o sufre una anomalía, se anula el registro del grupo del WLM.

Para implementar la optimización de la conexión DNS para solicitudes de IIOP (incluidas las solicitudes de enterprise beans), deben definirse las siguientes opciones de CORBASERVER:

- La opción HOSTNAME de la definición de CORBASERVER debe especificar un nombre de host genérico. Este de host genérico es el valor de DNSGROUP procedente de la definición de TCPIPSERVICE, con el sufijo del dominio o del subdomino gestionado por el servidor de nombres en MVS. Este nombre de dominio lo establece el administrador de TCP/IP. Así, en el ejemplo anterior, se podría utilizar WWW.PLEX1.IBM.COM para dirigir a CICSPROD1 y a CICSPROD2.

- El CORBASERVER con el nombre de host (o el DJARS que contiene) debe publicarse en el servidor de nombres.

El servidor de nombres debe estar configurado de forma que le permita buscar y resolver el nombre de host genérico.

Evitar problemas del Sistema de nombres de dominio (DNS): Importante

Para evitar dificultades al utilizar servidores de nombres, debe tener en cuenta lo siguiente:

- Las búsquedas de nombres dinámicos no se deben almacenar en memoria caché. Si utiliza un cliente que copie en caché los resultados de búsqueda de nombres de servidor, no puede tener la certeza de que sigue trabajando con la dirección IP correcta. Esto podría resultar en que el cliente intentase llamar continuamente a una región de servidor que se haya cerrado en lugar de obtener la dirección de otra región de servidor que haya adquiro del rol que antes cumplía el otro servidor.
- Se puede producir un problema debido a tensión en el servidor de nombres que se utiliza. Algunas búsquedas tienen éxito, otras fracasan con una `NameNotFoundException`.

  Cuando el número de búsquedas simultáneas es alto, quizá si un cliente o un bean realiza búsquedas repetidas sin copiar en caché, la posibilidad de encontrar uno de estos "errores" del servidor de nombres aumenta. Las medidas posibles a considerar son:
  - Instale una máquina de mayor capacidad para ejecutar el servidor de nombres.
  - Codifique las aplicaciones para que reconozcan esta posibilidad y que reintenten la operación cuando se encuentre este error.
  - Configure el sistema de MVS de forma que las direcciones utilizadas más habitualmente se incluyan en su archivo `/etc/hosts`. Con esto se elude la búsqueda de estos nombres en el servidor de nombres y se utiliza la dirección codificada en el archivo.
  - En lugar de especificar direcciones IP por nombre, especifíquelas por número. (Sin embargo, esta solución no es recomendable en un entorno de producción).

Programa de seguridad IIOP sustituible por el usuario

Este es un mecanismo de identificación opcional.

No es un mecanismo de autenticación, sino un modo de proporcionar un ID de usuario de CICS. Para utilizarlo, especifique el nombre del programa de seguridad en la opción URM de la definición TCPIPSERVICE para el puerto IIOP. Si lo hace, el procesador de solicitudes de IIOP llama a dicho programa de seguridad.

Al invocarlo, el programa de seguridad es preformateado con el valor definido por el parámetro de inicialización del sistema DFLTUSER (cuyo valor predeterminado es CICSUSER), pero puede sustituirlo. Antes del direccionamiento de la solicitud IIOP a un procesador de solicitudes, CICS comprueba con RACF que la transacción del receptor de peticiones tiene permitido lanzar un trabajo en nombre del ID de usuario generado por el programa de seguridad.

Puede escribir su propio programa para proporcionar un ID de usuario, o utilizar el programa de seguridad de ejemplo DFHXOPUS. Consulte "Utilización del programa de seguridad IIOP sustituible por el usuario" en la página 417.

Autenticación de CONNECTION

El ID de usuario del cliente se transmite desde la región de escucha a la AOR solo si se especifica ATTACHSEC(IDENTIFY) en la definición de CONNECTION en la AOR.

Consulte Link security with MR, en la *Guía de seguridad RACF de CICS*, para obtener más información.

A los usuarios de IIOP se les recomienda que especifiquen SEC=YES t ATTACHSEC(IDENTIFY).

Configuración de CICS para IIOP

Tiene que configurar CICS como un participante de CORBA para ejecutar todas las aplicaciones basadas en IIOP, incluidos los enterprise beans.

Además de los requisitos para ejecutar Java, también puede necesitar el siguiente software:

- Java Naming and Directory Interface (JNDI) versión 1.2.
- DB2 con controlador de servidor de datos IBM para ampliaciones JDBC y SQLJ.

Realice los pasos siguientes:

- "Configuración del sistema host para IIOP"
- "Configuración de TCP/IP para IIOP" en la página 407
- "Configuración de CICS para IIOP" en la página 409

Es posible que también tenga que realizar uno de estos pasos:

- "Configuración de un servidor LDAP" en la página 397
- "Configuración de un CORBA Object Services Naming Directory Server" en la página 407

Si elige "Configuración de un servidor LDAP" en la página 397, lea también "La estructura del espacio de nombres de LDAP" en la página 403.

Configuración del sistema host para IIOP

Para el soporte de IIOP, realice estas tareas del sistema:

Acerca de esta tarea

Procedimiento

1. Concesión a las regiones de CICS de acceso a z/OS UNIX System Services. Como parte de esta tarea, hará lo siguiente:
   a. Conceda a CICS acceso a los directorios y archivos z/OS UNIX que sean necesarios para crear JVM.
   b. Cree el directorio de trabajo de z/OS UNIX que haya especificado para entrada, salida y mensajes de la JVM y conceda a CICS acceso a los mismos.
2. "Configuración de JVM en agrupación" en la página 95. Durante esta tarea, hará lo siguiente:
   a. Permita a CICS localizar perfiles de JVM y cualquier archivo de propiedades de JVM asociado.

b. Elija los perfiles de JVM adecuados para su objetos sin estado CORBA y enterprise beans.
c. Si es necesario, personalice los perfiles de JVM para adecuarse a los requisitos de la región CICS. (En el curso de la configuración de CICS como un servidor CORBA, tendrá que añadir más información).

Al leer "Configuración de JVM en agrupación" en la página 95, recuerde que, *en el caso de objetos sin estado CORBA y enterprise beans*:

- El perfil de JVM utilizado es el que se especifica en la definición PROGRAM del programa **procesador de solicitudes**.
- Como sucede con todos los programas Java de CICS, si utiliza un archivo de propiedades de JVM, este se debe especificar en el perfil de JVM.
- El perfil de JVM predeterminado, que se especifica en la definición PROGRAM del programa procesador de solicitudes predeterminado, es DFHJVMCD.
- Si tiene planeado utilizar el perfil de JVM predeterminado con sus solicitudes de objeto sin estado CORBA y de enterprise bean, solo tiene que localizar DFHJVMCD y personalizar el perfil para la región CICS, como se describe en "Configuración de JVM en agrupación" en la página 95.

  Aunque tenga pensado utilizar perfiles de JVM personalizados, debe realizar los cambios en DFHJVMCD que sean necesarios para adaptarse a la configuración de la región CICS, ya que DFHJVMCD lo utiliza CICS internamente, además de su uso para el programa procesador de solicitudes predeterminado.

3. "Definición de un directorio de estantería". El directorio de estantería se utiliza para archivos JAR desplegados.
4. "Definición de servidores de nombres". Este paso es necesario solo si tiene que definir servidores de nombres para los fines descritos en este procedimiento.

Definición de un directorio de estantería:

Cada definición CORBASERVER debe especificar el nombre de un directorio de estantería en z/OS UNIX.

Cuando se instala una definición DJAR, copia el archivo JAR desplegado en un subdirectorio del directorio raíz de estantería. (Asimismo, cuando se emite un mandato PERFORM CORBASERVER PUBLISH, la IOR del CorbaServer se graba en el subdirectorio).

Puede darle a su directorio de estantería el nombre que desee. Sin embargo, se recomienda crearlo en algún lugar bajo el directorio `/var`. Por ejemplo, puede crear un directorio de z/OS UNIX llamado `/var/cicsts/`. Tras haber creado el directorio de estantería, debe conceder al ID de usuario de la región CICS acceso completo al mismo: lectura, grabación y ejecución. Consulte Concesión a las regiones de CICS de acceso a z/OS UNIX System Services para obtener orientación.

Definición de servidores de nombres:

Tal vez tenga que definir servidores de nombre para dos fines:

1. Si utiliza la optimización de conexión DNS (Sistema de nombres de dominio), las regiones de escucha se deben configurar para hablar con el mismo servidor de nombres en z/OS que el MVS Workload Manager esté configurado para utilizar.

Puede definir el servidor de nombres que TCP/IP debe utilizar proporcionando una sentencia SYSTCPD DD en el JVCL de inicio de CICS para la región de escucha, como se describe en Enabling TCP/IP in a CICS region , en el manual *Guía de instalación de CICS Transaction Server para z/OS*.

2. Una aplicación cliente puede localizar una aplicación de servidor de IIOP mediante referencias a objeto que se hayan registrado en el servidor de nombres. Por ejemplo, un cliente Java puede utilizar la interfaz JNDI para obtener una referencia a un objeto de aplicación de servidor como una instancia de la interfaz inicial de un enterprise bean. Las referencias a objeto se pueden registrar en un servidor de nombres desde CICS emitiendo los mandatos PERFORM CORBASERVER PUBLISH o PERFORM DJAR PUBLISH.

Habilitar referencias de JNDI:

Para permitir que sus aplicaciones obtengan referencias mediante una interfaz JNDI, configure un servidor de nombres que soporte la Java Naming and Directory Interface (JNDI) versión 1.2.

Puede utilizar cualquiera de las siguientes opciones:

Un servidor Lightweight Directory Access Protocol (LDAP)
Si se utiliza un servidor de nombres LDAP, los enterprise beans de CICS y WebSphere pueden interoperar más fácilmente en un espacio de nombres compartido. Consulte "Configuración de un servidor LDAP" en la página 397.

Un servicio de directorio de denominación Corba Object Services (COS).
Los COS Naming Servers se ejecutan en una máquina externa.

Se puede utilizar cualquier COS Naming Service estándar de la industria que soporte JNDI versión 1.2. Consulte "Configuración de un CORBA Object Services Naming Directory Server" en la página 407.

Especificación de la ubicación del servidor de nombres JNDI:

Para permitir que el código Java que se ejecuta en CICS emita llamadas a API de JNDI y que CICS publique referencias a las interfaces iniciales de enterprise beans o de IOR de objetos sin estado CORBA, debe definir la ubicación del servidor de nombres.

Acerca de esta tarea

Especifique la dirección web (URL) y el número de puerto TCP/IP del servidor de nombres utilizando la propiedad del sistema **-Dcom.ibm.cics.ejs.nameserver**. En "Propiedades del sistema de la JVM" en la página 118 puede encontrar información más detallada.

Importante:

1. Debe especificar la ubicación del servidor de nombres en la propiedad del sistema **-Dcom.ibm.cics.ejs.nameserver** en todos los perfiles de JVM o los archivos de propiedades que utilizan sus objetos CORBA sin estado o sus enterprise beans.
2. En concreto, asegúrese de especificar la ubicación del servidor de nombres en el perfil de JVM DFHJVMCD. El perfil DFHJVMCD lo utilizan los perfiles definidos por CICS, incluidos el programa procesador de solicitudes predeterminado y el programa que CICS utiliza para publicar y retirar archivos JAR desplegados.

3. También debe especificar la ubicación del servidor de nombres en cualquier otro perfil de JVM que elija para utilizar para objetos CORBA sin estado o enterprise beans. Pueden ser perfiles de JVM de ejemplo proporcionados por CICS o sus propios perfiles de JVM. En el caso de objetos sin estado CORBA y enterprise beans, los perfiles de JVM se indican en las definiciones de recurso PROGRAM para los programas procesadores de solicitudes.
4. Para obtener información detallada sobre la definición de la ubicación del servidor de nombres, consulte "Propiedades del sistema de la JVM" en la página 118.

Configuración de un servidor LDAP

Utilice un servidor LDAP configurado para WebSphere o configure uno nuevo.

Si dispone de un servidor LDAP ya existente para WebSphere:

Si el servidor de nombres que ha elegido para que lo utilice CICS ya se ha configurado para WebSphere Application Server for z/OS, probablemente necesitará muy poca configuración para permitir que CICS lo emplee.

La operación correcta del soporte de EJB en CICS requiere que el espacio de nombres LDAP se configure con un WebSphere System Namespace; los mecanismos de publicación y retirada de CICS intentan operar en una estructura de System Namespace. Sin embargo, una vez dentro de un método de EJB, o si se ejecuta una transacción habitual de Java en CICS, puede comunicarse con cualquier espacio de nombres de LDAP independientemente de si soporta un System Namespace.

Cuando se utiliza un servidor LDAP que no esté configurado con un WebSphere System Namespace, utilice un servicio de directorio alternativo, como el servicio LDAP proporcionado como parte del IBM Developer Kit for the Java Platform 5.0 base, en lugar de la fábrica de contexto de WebSphere que se incluye con CICS. Consulte "Fábrica de contexto LDAP suministrada con Java" en la página 315 para obtener detalles sobre cómo utilizar la fábrica LDAP.

Comprender la estructura de denominación de WebSphere que existe en el servidor LDAP (consulte "La estructura del espacio de nombres de LDAP" en la página 403) hace que sea más fácil para usted o para su administrador de LDAP determinar valores adecuados para las seis propiedades clave que debe conocer una región CICS. Estos se describen en "Propiedades del sistema de la JVM" en la página 118.Las tres propiedades de seguridad solo son necesarias si el espacio de nombres de LDAP está configurada de forma segura. En algunos servidores LDAP puede darse el caso de que todos los usuarios tengan acceso de grabación y que no sea necesario definir las propiedades del principal ni de las credenciales para la región CICS.

Si la estructura que WebSphere impone en el espacio de nombres es adecuada para sus necesidades, no es necesaria más configuración.

Los valores para nameserver, containerdn y noderootrdn se pueden obtener entendiendo la estructura del System Namespace y observando la estructura que hay en su servidor LDAP elegido. La última parte de esta sección analiza cómo determinar los valores de propiedad si se examina un espacio de nombres existente.

Razones para realizar más configuración:

Tal vez tenga que seguir con la configuración del servidor LDAP, incluso aunque el servidor ya esté configurado para WebSphere Application Server for z/OS, por cualquiera de estas razones.

1. La configuración de seguridad tiene que cambiarse para hacerse cargo a las regiones CICS que se introducen. Consulte "La estructura del espacio de nombres de LDAP" en la página 403 y "Consideraciones sobre la seguridad" en la página 405 para obtener más información sobre la estructura LDAP y los problemas de seguridad.
2. CICS tiene que ejecutarse en un *dominio* distinto a WebSphere. Si está creando un un dominio nuevo y distinto, WebSphere Application Server for z/OS y CICS no podrán localizar los enterprise beans del otro fácilmente. Sin embargo, si solo pretende compilar un nuevo dominio, los únicos pasos de configuración que tiene que ejecutar son Paso 4. "Creación del nodo legacyRoot" y Paso 5. "Aplicación de seguridad en el nivel de región CICS".
3. CICS tiene que ejecutarse en una estructura de espacio de nombres del sistema completamente distinta en el servidor LDAP. Es decir, CICS debe contar con un `containerdn` que apunte a un sitio distinto que la ubicación raíz del espacio de nombres del servidor. En ese caso, inicie el procedimiento de configuración en Paso 2. "Añada un nuevo sufijo". En ese caso, no es posible que los sistemas CICS y WebSphere Application Server for z/OS trabajen con los valores del contenedor que difieren para localizar los enterprise beans del otro.

Configuración de un nuevo servidor LDAP:

Si no tiene ningún servidor LDAP existente configurado para WebSphere Application Server for z/OS, realice estos pasos para configurar un nuevo servidor LDAP.

Acerca de esta tarea

1. Instale el esquema de denominación de WebSphere.
2. Añada un nuevo sufijo.
3. Cree el nodo raíz del espacio de nombres del sistema (containerdn).
4. Cree el nodo legacyRoot bajo el nodo raíz del espacio de nombres (noderootrdn).
5. Opcionalmente, aplique medidas de seguridad en el nivel de la región CICS.

Para realizar muchos de los pasos, es probable que necesite acceder a un principal de LDAP que tenga autoridad adecuada en el servidor LDAP para crear nuevas entradas en el nivel *raíz*.

Cuando haya completado estos pasos, puede determinar los valores de las propiedades del sistema necesarias en sus archivos de propiedades de JVM para permitir que CICS opere con el servidor LDAP y añadir dichas propiedades del sistema a todos los archivos de propiedades de JVM correspondientes.

Los pasos del siguiente ejemplo permiten configurar un servidor LDAP con los siguientes valores para las propiedades del sistema en los archivos de propiedades de JVM:

```
-Dcom.ibm.cics.ejs.nameserver=ldap://wibble.example.com:389
-Dcom.ibm.ws.naming.ldap.containerdn=ibm-wsnTree=t1,o=WASNaming,c=US
-Dcom.ibm.ws.naming.ldap.noderootrdn=ibm-wsnName=legacyRoot,ibm-wsnName=PLEX2,
      ibm-wsnName=domainRoots
-Djava.naming.security.authentication=simple
-Djava.naming.security.principal=cn=CICSSystems,c=US
-Djava.naming.security.credentials=secret
```

Se proporcionan valores similares para las propiedades del sistema de ejemplo en los archivos de propiedades de JVM de ejemplo proporcionados por CICS.

Un ejemplo:

En los archivos de configuración que se utilizan en este ejemplo hay notas que le orientarán sobre cómo adaptar este conjunto de propiedades a sus necesidades concretas.

La propiedad que es más probable que modifique es *noderootrdn*, ya que seguramente tenga algún dominio distinto a PLEX2 como agrupación para sus nodos. Este valor se especifica en el sistema en el Paso 4. “Creación del nodo legacyRoot”.

Tenga en cuenta que el ejemplo asume que existe un principal 'cn=admin' en el servidor LDAP, con la contraseña 'adminpwd', y que este principal tiene autorización para realizar cualquier operación en el servidor LDAP.

1. Instale el esquema de denominación de WebSphere.

   Si el servidor LDAP que se va a configurar ya tiene el esquema de denominación de WebSphere, puede omitir este paso. Un servidor de nombres LDAP configurado para WebSphere ya tendrá este esquema.

   Si se trata de cualquier otro servidor LDAP, instale el esquema de denominación de WebSphere. El esquema se incluye con CICS como `/usr/lpp/cicsts/cicsts42/utils/namespace/WebSphereNamingSchema.ldif` en z/OS UNIX.

   Nota: El archivo `WebSphereNamingSchema.ldif` necesita que antes se carguen `RFC2256.ldif` y `RFC2713.ldif`. Esto se debe a que la definición de la clase de objeto **ibm-wsnEntry** hace referencia al tipo de atributo **javaClassName**. Si se utiliza el servidor LDAP en z/OS, estos archivos LDAP de requisito previo no se cargan de forma predeterminada cuando se configura el servidor LDAP.

   El servidor LDAP en z/OS tiene que almacenar las entradas del esquema en el almacén de fondo al que se aplican. Esto se consigue añadiendo un sufijo al nombre distinguido de cada entrada del esquema. El archivo `WebSphereNamingSchema.ldif` que se proporciona no especifica ningún sufijo en las entradas del esquema, por lo que debe añadir uno. Por ejemplo, si el sufijo para el almacén de fondo es “`c=US`”, debe cambiar cada instancia de “`dn:cn=schema`” en el archivo `ldif` a “`dn:cn=schema,c=US`”.

   Aplique el esquema al servidor de nombres utilizando el mandato **ldapmodify**:

   ```
   ldapmodify -h <nombre_host>
              -p <número_puerto>
              -D <principal_autorizado>
              -w <contraseña_principal_autorizado>
              -f WebSphereNamingSchema.ldif
   ```

   Donde `nombre_host` y `número_puerto` son los correspondientes al servidor LDAP y el principal autorizado es el nombre distinguido de un usuario con autoridad suficiente en el servidor de nombres como para grabar entradas.

   El mandato **ldapmodify** debe estar disponible para el servidor LDAP que elija. Si no lo está, consulte la documentación de su servidor LDAP para determinar cómo se debe instalar un nuevo esquema (en formato ldif).

   Un ejemplo específico podría ser:

```
ldapmodify -h wibble.example.com
           -p 389
           -D cn=admin
           -w adminpwd
           -f WebSphereNamingSchema.ldif
```

2. Añada un nuevo sufijo.

   Para compilar una nueva jerarquía en el espacio de nombres es necesario crear un nuevo sufijo de nombre distinguido base. En esta configuración de ejemplo, el sufijo es *c=US*, y la nueva jerarquía va a ser *ibm-wsnTree=t1,o=WASNaming,c=US*. El procedimiento para añadir un sufijo varía según los proveedores de LDAP. Su documentación de LDAP debe indicar cómo hacerlo para el proveedor que haya elegido. Como ejemplo, aquí tiene el procedimiento para añadir un sufijo a una instalación de z/OS Communications Server en Windows 32:

   - Lance la interfaz de administración de LDAP en un navegador web escribiendo `http://[nombre_host]/ldap`, donde nombre_host es el nombre de host de la máquina en la que está instalado el directorio de LDAP. Se visualiza la ventana de inicio de sesión de administración.
   - Escriba el ID de usuario (por ejemplo, con el formato cn=root) y la contraseña del administrador.
   - Asegúrese de que el servidor LDAP se está ejecutando.
   - En el panel de navegación izquierdo, pulse en la carpeta Settings (valores) y luego pulse en Suffixes (sufijos).
   - Escriba el nombre del DN base que utilizar como sufijo (en nuestro ejemplo, "c=US") y pulse en Update (actualizar).
   - Tras añadir el sufijo del DN base, detenga y reinicie el servidor LDAP.

   Ahora el sufijo ya existe en su sistema LDAP.

   En un sistema z/OS, actualice el archivo `slapd.conf` para especificar el nuevo sufijo en el sistema, y luego reinicie el servidor de nombres. La línea extra para añadir a `slapd.conf` es:

   ```
   suffix "c=US"
   ```

3. Cree el nodo raíz del espacio de nombres del sistema (containerdn)

   En CICS se proporciona un archivo ldif para compilar la raíz del espacio de nombres del sistema (un nodo denominado containerdn), en `utils/namespace/dfhsns.ldif`. Este archivo contiene comentarios que describen cómo adaptarlo a su entorno. Si se utiliza sin modificarlo, crea un containerdn de `ibm-wsnTree=t1,o=wasnaming,c=US` y también dos usuarios de CICS en el espacio de nombres de LDAP. El primer usuario de CICS tiene un nombre distinguido de `cn=CICSSystems,c=US` y el segundo es `cn=CICSUser,c=US`.

   Se definen dos ID de usuario. Para comprender cómo se utilizan, consulte “Consideraciones sobre la seguridad” en la página 405.

   El mandato ldapmodify debe estar disponible para el servidor LDAP que haya elegido. Si no lo está, consulte la documentación de su servidor LDAP para determinar cómo se debe compilar la raíz del espacio de nombres del sistema.

   Este archivo LDIF se puede aplicar al servidor LDAP del siguiente modo:

   ```
   ldapmodify-h <nombre_host>
          -p <número_puerto>
          -D <principal_autorizado>
          -w <contraseña_principal_autorizado>
          -f dfhsns.ldif
   ```

Donde nombre_host y número_puerto son los correspondientes al servidor LDAP y el principal autorizado es el nombre distinguido de un usuario con suficiente autoridad en el servidor de nombres como para grabar entradas.

Un ejemplo específico es:

```
ldapmodify-h wibble.example.com
        -p 389
        -D cn=admin
        -w adminpwd
        -f dfhsns.ldif
```

4. Compile el nodo legacyRoot bajo el nodo raíz del espacio de nombres (noderootrdn)

   El nodo legacyRoot del espacio de nombres es el punto en el que CICS normalmente está configurado para situarse cuando es llamado para crear un nuevo InitialContext. Para este paso, el script `DFHBuildSNS` se incluye con CICS en el directorio `utils/namespace`.

   La sintaxis es :

```
DFHBuildSNS   -ldapserver  <url_servidor>
             [-node        <nodo en el dominio>]
              -domain      <nombre_dominio>
              -containerdn <Raíz del espacio de nombres>
              -principal   <principal autorizado para grabar en el espacio de nombres>
              -credentials <contraseña para ese principal>
             [-force]
```

   Por ejemplo:

```
DFHBuildSNS    -ldapserver ldap://wibble.example.com:389
               -domain PLEX2
               -containerdn ibm-wsnTree=t1,o=WASNaming,c=US
               -principal cn=admin
               -credentials adminpwd
```

   La opción *-force* solo se utiliza con el distintivo *-node*, pero nunca se emplea en un entorno CICS.

5. Como opción, puede aplicar las medidas que se describen en “Seguridad en el nivel de la región CICS” en la página 405.

Tras ejecutar este script, pueden determinarse los valores de las propiedades del sistema necesarias en sus archivos de propiedades de JVM y puede añadirlas todas a los archivos de propiedades de JVM correspondientes.

Determinación de los valores para las propiedades del sistema LDAP:

Estas propiedades del sistema están relacionadas con la utilización de un espacio de nombres LDAP para JNDI.

En “Propiedades del sistema de la JVM” en la página 118 se muestran descripciones completas de cada una de estas propiedades del sistema.

- Si acaba de configurar este espacio de nombres LDAP, ya conoce los valores que ha utilizado. Algunos de estos son necesarios para definir las propiedades de CICS.
- Si utiliza o reutiliza un espacio de nombres del sistema ya existente, pregunte al administrador de LDAP los valores adecuados para dichas propiedades.
- Si no tiene acceso al administrador de LDAP o los valores no están disponibles, tal vez pueda determinarlos con la ayuda de la siguiente información examinando el espacio de nombres.

No es probable que descubra el principal de seguridad ni las credenciales examinando el espacio de nombres.

-Dcom.ibm.cics.ejs.nameserver
Es el URL para el servidor LDAP que se configura. En el ejemplo de "Configuración de un nuevo servidor LDAP" en la página 398, es *ldap://wibble.example.com:389*

-Dcom.ibm.ws.naming.ldap.containerdn
Es el valor especificado en el archivo `dfhsns.ldif`. El valor predeterminado es *ibm-wsnTree=t1,o=WASNaming,c=US* si no modificó el archivo ldif. Si busca esta valor examinando el espacio de nombres ya existente, busque un nodo del tipo ibm-wsnTree; la vía de acceso hasta este nodo es un valor posible para containerdn.

-Dcom.ibm.ws.naming.ldap.noderootrdn
Se puede determinar a partir del dominio que se especificara en la llamada a DFHBuildSNS. En el ejemplo, el noderootrdn es *ibm-wsnName=legacyRoot,ibm-wsnName=PLEX2,ibm-wsnName=domainRoots*. Si busca esta valor examinando un espacio de nombres ya existente, busque la vía de acceso desde el containerdn elegido para la entrada legacyRoot.

-Djava.naming.security.authentication
Se define en *simple* si CICS debe autenticarse ante LDAP para enlazar con él (o grabar en él). Si se utilizan los valores predeterminados de los scripts proporcionados, la autenticación es necesaria porque el script `dfhsns.ldif` eliminó el acceso de grabación predeterminado para el grupo ANYBODY y le otorgó acceso de grabación al nuevo principal `cn=CICSUser,c=US` que creó. Si CICS no tiene que autenticarse ante LDAP para grabar en él, no defina ningún valor para esta propiedad del sistema.

Importante: Si se especifica esta propiedad del sistema, también debe especificar **-Djava.naming.security.principal** y **-Djava.naming.security.credentials**. Dado que estas propiedades del sistema contienen el ID de usuario y la contraseña que CICS necesita para acceder al servicio LDAP seguro, debe prestar especial atención a los controles de acceso en vigor en su instalación para los archivos que contienen estas propiedades del sistema. Debe asegurarse de que los archivos sean seguros, con la autorización de actualización restringida a los administradores del sistema.

-Djava.naming.security.principal
Es un principal con la autoridad para enlazar con el espacio de nombres. Tal vez elija el principal del sistema que tenga acceso de grabación a todo el espacio de nombres si la seguridad no es una preocupación real. Sin embargo, le aconsejamos que utilice al menos el nombre distinguido *cn=CICSUser,c=US* especificado en `dfhsns.ldif`, porque ese identificador solo puede grabar en un área concreta del espacio de nombres LDAP (el containerdn y debajo).

Si quiere una seguridad aún más restringida, el principal puede ser *cn=CICSSystems,c=US*. Si utiliza este ID, debe realizar una configuración de LDAP extra. Consulte "Consideraciones sobre la seguridad" en la página 405 para obtener un análisis completo de la configuración de seguridad LDAP de CICS.

-Djava.naming.security.credentials
Es la contraseña para el principal. El valor predeterminado si no adaptó `dfhsns.ldif` es *secret*.

Cuando haya determinado los valores de estas propiedades del sistema, especifíquelos en todos los perfiles de JVM o los archivos de propiedades de JVM opcionales que utilizan las aplicaciones CORBA o los enterprise beans.

En concreto, especifíquelos en el perfil de JVM DFHJVMCD o el archivo de propiedades de referencia. El perfil DFHJVMCD lo utilizan los perfiles definidos por CICS, incluidos el programa procesador de solicitudes predeterminado y el programa que CICS utiliza para publicar y retirar archivos JAR desplegados.

También debe especificar estas propiedades del sistema en los perfiles de JVM o los archivos de propiedades a los que haga referencia cualquier otro perfil de JVM que elija para utilizar para objetos CORBA sin estado o enterprise beans. Estos perfiles pueden ser perfiles de JVM de ejemplo proporcionados por CICS o sus propios perfiles de JVM. En el caso de objetos sin estado CORBA y enterprise beans, los perfiles de JVM se indican en las definiciones de recurso PROGRAM para los programas de procesador de solicitudes.

La estructura del espacio de nombres de LDAP

La estructura del espacio de nombres de LDAP que utiliza WebSphere Application Server versión 4 para z/OS es una estructura cómoda para utilizar en un entorno de CICS.

Nota: WebSphere Application Server versión 5 y posteriores utilizan un COS Naming Server de forma predeterminada y soportan LDAP únicamente para la compatibilidad con versiones anteriores de WebSphere Application Server versión 4.

Hay dos nodos importantes en la estructura del espacio de nombres de LDAP que utiliza WebSphere, la raíz de contenedor (container) y la raíz heredada (legacy).

La raíz del contenedor (container root):

La raíz del contenedor es un nodo de tipo ibm-wsnTree. De forma predeterminada, se llama *ibm-wsnTree=t1, o=wasnaming, c=us*. Sin embargo, esto se puede personalizar cambiando el archivo *bboldif.cb* incluido con WebSphere.

La raíz heredada (legacy root):

La raíz heredada es un nodo de tipo *ibm-wsnName* situado en algún punto bajo la raíz del contenedor.

Un nombre típico para el mismo puede ser: *ibm-wsnName=legacyRoot,ibm-wsnName=PLEX2,ibm-wsnName=domainRoots,ibm-wsnTree=t1,o=WASNaming,c=us*. Los nombres legacyRoot y domainRoots son fijos. La única variable es el nombre intermedio, que en este ejemplo es PLEX2.

Puede haber varios nodos legacyRoot, cada un con un nombre distinto. Cada uno de ellos es un "dominio". La configuración de WebSphere Application Server for z/OS correlaciona un dominio con un sysplex. Se configura cuando se especifica el nombre del sysplex en el diálogo de personalización, al instalar WebSphere Application Server for z/OS.

Dominios:

Un dominio contiene varios servidores.

En WebSphere Application Server for z/OS, cada servidor cuenta con un nodo bajo legacyRoot; por ejemplo, un servidor llamado BBOSV1 tendría un nombre *ibm-wsnName=BBOSV1,ibm-wsnName=PLEX2* relativo a la raíz heredada y los objetos que publique estarán bajo este nodo.

Cuando se configura CICS para utilizar el mismo servidor LDAP que WebSphere, cada CorbaServer de CICS tiene un nodo justo bajo legacyRoot. Por lo tanto, si un CorbaServer tiene un prefijo JNDI de CICS1, habrá un nodo *ibm-wsnName=CICS1* relativo a la raíz heredada y CICS publicará los objetos del CorbaServer bajo este nodo. Si se crea un nuevo contexto inicial (InitialContext) en WebSphere Application Server for z/OS, o en CICS configurado como se indica anteriormente, InitialContext se basará en el nodo legacyRoot. Esto facilita a los enterprise beans de CICS buscar objetos publicados por WebSphere y a los enterprise beans o los servlets de WebSphere buscar objetos publicados por CICS.

Nota: Cualquier subcontexto JNDI bajo el contexto JNDI inicial de la región CICS (que generalmente es el nodo legacyRoot) puede ser transitorio. Este es el caso si CICS tiene acceso de grabación al nodo de contexto inicial.

En la opción JNDIPREFIX de la definición de CORBASERVER se especifica un subcontexto JNDI del CorbaServer. CICS crea el subcontexto (si tiene el permiso de grabación necesario y no existe ya el subcontexto en la estructura del espacio de nombres) cuando se publica un enterprise bean desde el CorbaServer. Sin embargo, si se retiran todos los enterprise beans del CorbaServer, CICS puede suprimir el subcontexto desde la estructura del espacio de nombres. Si varios CorbaServer comparten parte de la jerarquía de prefijos, CICS nunca elimina los contextos que todavía esté utilizando alguno de ellos. Pero si los contextos del prefijo están vacíos, se eliminan ya en el contexto inicial.

Si desea evitar que se suprima el nodo de nivel superior de la jerarquía del subcontexto, no proporcione a CICS acceso de grabación al nodo de contexto inicial. (Esto significa que debe crear el nodo de nivel superior del subcontexto manualmente). Si desea proteger varios niveles superiores de la jerarquía del subcontexto, otorgue a CICS permiso de grabación solo para los niveles inferiores. (Esto significa que debe crear los nodos de nivel superior del subcontexto manualmente). Para obtener más información, consulte “Seguridad en el nivel de la región CICS” en la página 405.

Las versiones de WebSphere Application Server para plataformas distribuidas tienen un concepto similar de dominio, pero dicho concepto no está relacionado con un sysplex.

Nodos:

Hay otro concepto, el de ***nodo***. Un dominio representa un número de nodos, y puede navegar hasta un dominio si conoce el nombre de nodo en lugar del nombre de dominio. Así, el nodo es una especie de alias para un dominio.

Los nodos se utilizan en versiones de WebSphere Application Server para plataformas distribuidas, pero no en WebSphere Application Server para z/OS. CICS no los utiliza. Sin embargo, parte de la estructura para el soporte de nodos se crea al configurar un nuevo servidor LDAP para que lo utilice CICS. Como WebSphere Application Server para z/OS no utiliza nodos, nodename es un parámetro opcional para el programa de utilidad DFHBuildSNS, que en CICS crea el espacio de nombres del sistema.

Consideraciones sobre la seguridad: Si se especificó que CICS debe autenticarse a sí mismo ante LDAP para grabar en él, mediante la codificación de la propiedad del sistema **-Djava.naming.security.authentication=***simple* en los perfiles de JVM, ahora tiene las siguientes opciones:
- "Seguridad en el nivel de containerdn", o
- "Seguridad en el nivel de la región CICS".

Para ayudarse a decidir, en la Figura 37 se muestra una vista muy simplificada del espacio de nombres LDAP.

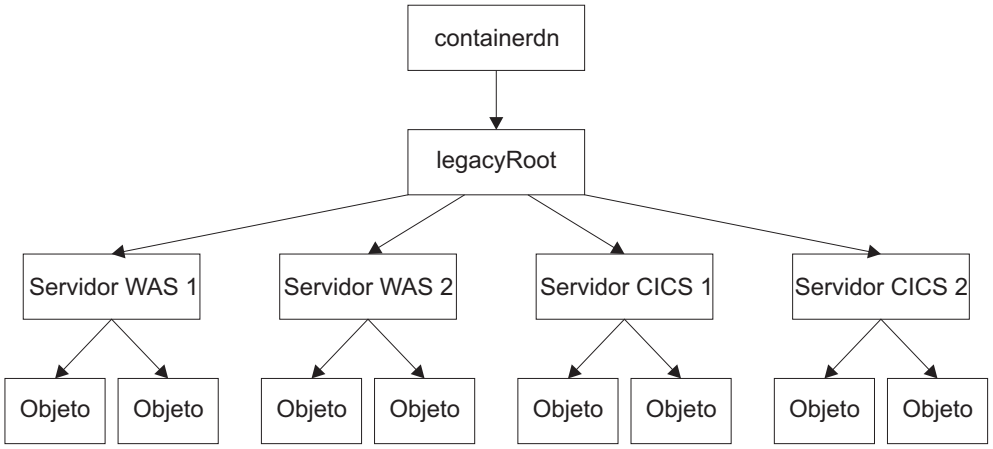


Figura 37. Vista simplificada de parte de un espacio de nombres LDAP

Si utiliza seguridad en el nivel de containerdn, CICS tiene acceso de grabación a containerdn y a todos los nodos bajo este. Esto permite a CICS, o a una aplicación de CICS que utilice interfaces JNDI, grabar en todos estos nodos, incluidos aquellos que pertenecen a WebSphere Application Server for z/OS. Si utiliza seguridad en nivel de región CICS, CICS y las aplicaciones CICS solo pueden grabar en los nodos CICS específicos del árbol.

Seguridad en el nivel de containerdn: Para utilizar seguridad en el nivel de containerdn, utilice el principal de administración de CICS (cn=CICSUser,c=us) creado por el archivo *dfhsns.ldif* consulte Paso 3. "Compile el nodo raíz del espacio de nombres del sistema". Otorgue a este principal acceso al nodo containerdn cuando lo cree. Asegúrese de que que este ID de usuario y su contraseña aparecen en las propiedades del sistema **-Djava.naming.security.principal** y **-Djava.naming.security.credentials**, en el archivo de propiedades de la JVM.

Seguridad en el nivel de la región CICS:

Otorgue a este principal acceso al nodo containerdn cuando lo cree. Asegúrese de que que este ID de usuario y su contraseña aparecen en las propiedades del sistema **-Djava.naming.security.principal** y **-Djava.naming.security.credentials**, en el archivo de propiedades de la JVM.

Para utilizar seguridad en el nivel de la región CICS, utilice el principal de tiempo de ejecución de CICS (cn=CICSSystems,c=US) creado por el archivo *dfhsns.ldif*; consulte Paso 3. "Compile el nodo raíz del espacio de nombres del sistema". Esto implica realizar algunos pasos más. Asegúrese de que que este ID de usuario y su contraseña aparecen en las propiedades del sistema **-Djava.naming.security.principal** y **-Djava.naming.security.credentials**, en el archivo de propiedades de la JVM. Además, como CICS no cuenta con acceso de

grabación para legacyRoot, no podrá crear su propio nodo (llamado Servidor de CICS 1 en la Figura 37 en la página 405), por lo que deberá hacerlo de forma manual y conceder al principal de tiempo de ejecución de CICS (cn=CICSSystems,c=US) acceso de grabación a dicho nodo. Esto se describe más adelante.

Para configurar una región CICS de este modo y luego utilizar el nuevo subcontexto:

- Elija un subcontexto adecuado, al que llamaremos *cicsabcd*.
- Cree dicho subcontexto bajo el legacyRoot que utilizará un sistema CICS (consulte "Creación de un subcontexto").
- Asegúrese de que el principal de tiempo de ejecución de CICS puede grabar en él.
- Especifique el principal y las credenciales de tiempo de ejecución de CICS mediante las propiedades del sistema **-Djava.naming.security.principal** y **-Djava.naming.security.credentials**, en los archivos de propiedades de JVM que se utilizan en la región.
- Asegúrese de que las definiciones CORBASERVER creadas en la región CICS tengan atributos JNDIPREFIX que comiencen por *cicsabcd*. Esto significa que las referencias que publiquen lo harán *debajo* del nuevo subcontexto *cicsabcd* bajo legacyRoot.

Ya se ha completado la configuración de la seguridad. Un usuario que examine el espacio de nombres LDAP puede localizar este contexto *cicsabcd* bajo legacyRoot y relacionarlo con las definiciones CORBASERVER.

Creación de un subcontexto: Para crear el subcontexto *cicsabcd* bajo legacyRoot en el espacio de nombres de LDAP y definir las lista de control de accesos (ACL) adecuadas para él, utilice el archivo LDIF proporcionado con CICS en *utils/namespace/dfhNewCICSSubcontext.ldif*.

- El archivo LDIF contiene comentarios para explicar los pasos implicados y los valores que es probable que haya que alterar para la configuración de un espacio de nombres del sistema LDAP concreto.
- El archivo LDIF se puede aplicar al servidor LDAP utilizando el mandato ldapadd:

  ```
  Ldapadd -h wibble.example.com
          -p 389
          -D cn=CICSUser,c=us
          -w CICSUserpwd
          -f dfhNewCICSSubcontext.ldif
  ```

  donde `CICSUserpwd` es la contraseña para el `CICSuser` (usuario de CICS) establecido al configurar `CICSuser`.

  Este mandato tiene que ejecutarse con un principal (y credenciales) que pueda grabar en el nodo legacyRoot. En el ejemplo que estamos utilizando, es *cn=CICSUser,c=US id*, que se ha creado para esta finalidad.
- La línea más importante del archivo LDIF es el cambio del nombre distinguido del nodo que se crea. Si asumimos que el espacio de nombres del sistema LDAP se ha configurado utilizando los scripts predeterminados incluidos con CICS, el nombre distinguido es:

  ```
  ibm-wsnName=cicsabcd,ibm-wsnName=legacyRoot,ibm-wsnName=PLEX2,
  ibm-wsnName=domainRoots,ibm-wsnTree=t1,o=wasnaming,c=US
  ```
- El resto del LDIF define las listas de control de accesos de forma adecuada para el nuevo nodo.

- Los comentarios de este archivo LDIF son importantes, ya que explican otras cosas que pueda que sea necesario tener en cuenta. Por ejemplo, tal vez haya algunas entradas adicionales de la ACL que sean apropiadas para su instalación en función de qué principales tengan actualmente acceso de grabación al espacio de nombres del sistema.
- Una vez aplicado el LDIF, el nuevo nodo existe en el servidor LDAP bajo el legacyRoot, y las listas de control de accesos están definidas de forma que el principal de tiempo de ejecución de CICS tenga acceso de grabación.

Otras consideraciones: Tal vez quiera tener en cuenta los siguientes aspectos:

- Podría crear varios principales distintos de tiempo de ejecución de CICS para distintas regiones, y así reducir el ámbito del acceso otorgado a cada principal.
- Si utiliza este proceso en un espacio de nombres del sistema ya existente, pueden estar utilizándose otros principales (y otras credenciales). Tienen que recibir acceso de grabación al nuevo subcontexto creado por *dfhNewCICSSubcontext*. Los comentarios del archivo LDIF dfhNewCICSSubContext analizan formas de comprobar si esto es así y cómo adaptar el archivo LDIF de forma adecuada antes de ejecutar el ldapadd.

Configuración de un CORBA Object Services Naming Directory Server

El modo más cómodo de configurar un CORBA Object Services Naming Directory Server es utilizar IBM WebSphere Application Server ejecutándose en una máquina Windows externa.

Acerca de esta tarea

El modo más cómodo de configurar un CORBA Object Services Naming Directory Server es utilizar IBM WebSphere Application Server ejecutándose en una máquina Windows externa. Siga las instrucciones de instalación que se incluyen con el mismo.

Configuración de TCP/IP para IIOP

Para configurar una región CICS como región de escucha de TCP/IP y que acceda a solicitudes de IIOP y las envíe, tiene que realizar las siguientes definiciones en CICS.

Acerca de esta tarea

1. En la secuencia de trabajos de inicio de CICS para todas las regiones CICS en las que sea necesaria la región de escucha, defina los siguientes parámetros de inicialización del sistema:
   - **IIOPLISTENER** como **YES**
   - **TCPIP** como **YES**
2. Defina e instale las definiciones de recurso TCPIPSERVICE en la región de escucha para cada puerto que la región de escucha supervisará, especificando:
   - PROTOCOL(IIOP)
   - El puerto o la dirección IP en la que CICS escuchará las solicitudes de IIOP entrantes.

     Nota: Si la conexión SSL tiene una anomalía, algunos clientes tratarán de reintentar la operación en un puerto asociado que no sea SSL. CICS TS define este puerto para que sea el puerto SSL–1. Debe asegurarse de que este puerto (puerto SSL–1) no está definido para ningún otro fin. Los puertos IIOP bien conocidos son 683 (no SSL) y 684 (SSL).

- La transacción de CICS que se debe iniciar cuando llegue una solicitud. Para un servicio de IIOP, debe definirse como el receptor de peticiones IIOP de CICS, CIRR.
- El nivel de autenticación SSL (capa de sockets seguros) que utilizar.
- El nombre del DNSGROUP si se va a utilizar la optimización de conexión DNS. Consulte "Definición de recurso para optimización de la conexión DNS" en la página 392.
- El nombre del programa sustituible por el usuario que llamar para asociar esta solicitud con un ID de usuario de CICS para fines de gestión de la seguridad o de la carga de trabajo. Si se omite, no se llama a ningún programa sustituible por el usuario. Se proporciona un programa sustituible por el usuario de ejemplo, DFHXOPUS: consulte "Utilización del programa de seguridad IIOP sustituible por el usuario" en la página 417.

Por ejemplo:

```
DEFINE TCPIPSERVICE(IIOPNSSL) GROUP(DFH$IIOP)
     DESCRIPTION(IIOP TCPIPSERVICE sin soporte de SSL)
     URM(DFHXOPUS)          BACKLOG(10)            PORTNUMBER(683)
     TRANSACTION(CIRR)      SSL(NO)
     STATUS(CLOSED)         PROTOCOL(IIOP)
```

Importante: En un servidor multirregión, las definiciones de TCPIPSERVICE se deben instalar en *todas* las regiones (tanto de escucha como AOR) del servidor lógico. En las regiones de escucha, el parámetro de inicialización del sistema IIOPLISTENER se debe definir como 'YES'. En las AOR, se debe definir como 'NO'. En una combinación de región de escucha/AOR, se debe definir como 'YES'.

Consulte Recursos TCPIPSERVICE en la Guía de definición de recursos para obtener la sintaxis completa de la definición del recurso TCPIPSERVICE.

Utilización de la optimización de la conexión DNS:

Para utilizar la optimización de la conexión DNS con IIOP, tiene que definir un nombre de DNSGROUP en la definición de recurso IIOP TCPIPSERVICE.

Todas las regiones CICS que proporcionan el mismo TCPIPSERVICE con el mismo nombre de DNSGROUP se registran en MVS Workload Management (WLM) con el mismo *group-name* (nombre de grupo), como candidatos para solicitudes de cliente que necesiten el mismo servicio. Este registro también incluye el *nombre de host* de la región, que se obtiene mediante la función **gethostbyaddr** de TCP/IP, y un *nombre de servidor* exclusivo, que CICS obtiene del ID de aplicación genérico de la región, como especifica el parámetro de inicialización del sistema APPLID.

Las regiones de escucha se deben configurar para hablar con el mismo servidor de nombres DNS en z/OS que el MVS Workload Manager esté configurado para utilizar. Puede definir el servidor de nombres que TCP/IP debe utilizar proporcionando una sentencia SYSTCPD DD en el JVCL de inicio de CICS, como se describe en Habilitación de TCP/IP en una región CICS, en la *Guía de instalación de CICS Transaction Server para z/OS*.

Nota:

1. Tanto el cliente como el servidor de CICS tienen que utilizar el mismo servidor de nombres TCP/IP.
2. El servidor de nombres tiene que poder llevar a cabo una búsqueda inversa, es decir, tiene que poder convertir la dirección IP del servidor en un nombre de host completo.

Configuración de CICS para IIOP

Para el soporte de IIOP, debe definir una secuencia de trabajos de inicio de CICS y definir e instalar algunos recursos de CICS.

Definición de la secuencia de trabajos de inicio de CICS:

Debe definir los parámetros en la secuencia de trabajos de inicio para una región CICS que soporte IIOP:

Parámetro de JCL

REGION
Se recomienda un mínimo de 1000 M.

Parámetros de inicialización del sistema CICS

EDSALIM
Se recomienda un mínimo de 500 M.

IIOPLISTENER

- Especifique IIOPLISTENER=YES si la región CICS es una región de escucha de IIOP o una combinación de región de escucha y de región propietaria de la aplicación (AOR).
- Especifique IIOPLISTENER=NO si la región CICS es una región propietaria de la aplicación. Las definiciones de TCPIPSERVICE instaladas en la región que especifican PROTOCOL(IIOP) no se pueden abrir.

JVMPROFILEDIR
Defina en el directorio de z/OS UNIX que contiene los perfiles de JVM que esté utilizando para sus aplicaciones. En "Definición de la ubicación de los perfiles de JVM" en la página 81 se explica cómo realizar esta tarea.

KEYRING
Necesario si se utiliza autenticación de capa de sockets seguros (SSL) con certificados registrados en RACF.

MAXJVMTCBS
Especifique el número de JVM que la región CICS puede soportar. En "Gestión de la agrupación de JVM para mejorar el rendimiento" en la página 172 se describe cómo obtener un valor adecuado para el parámetro de inicialización del sistema **MAXJVMTCBS**.

TCPIP Defina como YES.

Sentencias DD para conjuntos de datos de CICS
Las definiciones de conjunto de datos VSAM locales de ejemplo se suministran en el grupo de RDO DFHEJVS proporcionado por CICS. Estos conjuntos de datos deben autorizarse con RACF para el acceso de UPDATE. Consulte Authorizing access to CICS data sets, en la *Guía de seguridad RACF de CICS*.

DFHEJDIR
Un archivo compartido recuperable que contiene el directorio de secuencias de solicitud. Este puede ser un archivo VSAM o una tabla de datos de recurso de acoplamiento. CICS proporciona JCL de ejemplo para ayudarle a crear este archivo, en el miembro DFHDEFDS de la biblioteca SDFHINST.

Nota: En la mayoría de los casos, el parámetro RECORDSIZE del JCL proporcionado no debería requerir ninguna modificación. Sin embargo,

si pretende instalar más de 40 CorbaServers en el servidor EJB/CORBA lógico, consulte "Especificación del RECORDSIZE de DFHEJDIR y de DFHEJOS".

DFHEJOS
Un archivo compartido no recuperable utilizado por CICS cuando se instalan CorbaServers para almacenar beans de sesión con estado que se han desactivado. Este puede ser un archivo VSAM o una tabla de datos de recurso de acoplamiento. CICS proporciona JCL de ejemplo para ayudarle a crear este archivo, en el miembro DFHDEFDS de la biblioteca SDFHINST.

Nota: En la mayoría de los casos, el parámetro RECORDSIZE del JCL proporcionado no debería requerir ninguna modificación. Sin embargo, si pretende instalar más de 40 CorbaServers en el servidor EJB/CORBA lógico, consulte "Especificación del RECORDSIZE de DFHEJDIR y de DFHEJOS".

Especificación del RECORDSIZE de DFHEJDIR y de DFHEJOS:

El número máximo de CorbaServer que se pueden definir para un servidor lógico EJB/CORBA de CICS se controla mediante los valores de RECORDSIZE del archivo de directorio de las secuencias de solicitud, de DFHEJDIR y del archivo de almacén de objetos EJB, DFHEJOS.

Los atributos de RECORDSIZE en las definiciones proporcionadas de JCL y de FILE para DFHEJDIR especifican un RECORDSIZE de 1017 bytes. Los atributos de RECORDSIZE en las definiciones proporcionadas de JCL y de FILE para DFHEJOS especifican un RECORDSIZE de 8185 bytes. Normalmente, no debe ser necesario modificar estos valores. Únicamente es necesario cambiarlos si pretende instalar más de 40 CorbaServer en el servidor lógico de EJB/CORBA.

Tanto DFHEJDIR como DFHEJOS contienen un registro de control que está compuesto de una cabecera de 24 bytes y de un grupo repetido de campos de control de CorbaServer, cada uno de 24 bytes de largo. La longitud predeterminada de 1017 para DFHEJDIR limita de forma efectiva el servidor lógico a 41 CorbaServer: (1 + 41) * 24 = 1008 bytes. Si necesita instalar un número de CorbaServer mayor que este en el servidor lógico, calcule el RECORDSIZE necesario para DFHEJDIR de este modo:

1. Multiplique el número necesario de CorbaServer por 24.
2. Añada 24 bytes para la cabecera del registro de control. Esto proporciona el tamaño mínimo absoluto de registro.
3. Redondee el último valor hasta el siguiente múltiplo de 512 para obtener el tamaño mínimo de intervalo de control.
4. Reste 7 para obtener el valor para el parámetro RECORDSIZE.

Haga que el valor de RECORDSIZE para DFHEJOS sea mayor que el de DFHEJDIR. Una longitud muy corta producirá colisiones al desactivar beans. (Las definiciones proporcionadas hacen que el RECORDSIZE de DFHEJOS sea casi 8 veces el de DFHEJDIR).

Nota: El JCL de ejemplo para DFHEJDIR y DFHEJOS está en el miembro DFHDEFDS de la biblioteca SDFHINST. Las definiciones del recurso FILE para DFHEJDIR y DFHEJOS están en el grupo de RDO de DFHEJVR, con definiciones FILE del recurso de acoplamiento de ejemplo en el grupo DFHEJCF y definiciones FILE de ejemplo de VSAM RLS en el grupo DFHEJVR.

Definición de recursos de CICS:

Debe crear los recursos de CICS necesarios para enterprise beans.

FILE

Proporcione e instale definiciones de recurso FILE para los siguientes archivos que CICS necesita:

El "Directorio de EJB", DFHEJDIR
: Un archivo que contiene un directorio de secuencias de solicitudes; el directorio se utiliza en el direccionamiento de solicitudes de método para enterprise beans y objetos sin estado CORBA. Debe definir DFHEJDIR como recuperable.

El "Almacén de objetos EJB", DFHEJOS
: Un archivo de beans de sesión con estado que se hayan desactivado. También se utiliza cuando se instalan CorbaServers. Debe definirlo como no recuperable.

En un servidor EJB/CORBA de CICS de región única, es aceptable definir DFHEJDIR y DFHEJOS como archivos locales. Sin embargo, en el caso de un servidor EJB/CORBA de CICS multirregión:

- DFHEJDIR deben compartirlo todas las regiones del servidor (tanto regiones de escucha como AOR).
- DFHEJOS deben compartirlo todas las AOR del servidor.

Para permitir que DFHEJDIR y DFHEJOS sean compartidos en varias regiones, puede definirlos de una de estas formas:

- Como archivos remotos en una región propietaria del archivo (FOR).
- Como tablas de datos de recursos de acoplamiento.
- Mediante el uso de compartición a nivel de registro VSAM (VSAM RLS).

Hay definiciones FILE de ejemplo para DFHEJDIR y DFHEJOS en el grupo de RDO proporcionado por CICS, DFHEJVS. Hay definiciones FILE de recurso de acoplamiento para DFHEJDIR y DFHEJOS en el grupo de RDO proporcionado por CICS, DFHEJCF. Hay definiciones FILE de VSAM RLS para DFHEJDIR y DFHEJOS en el grupo de RDO proporcionado por CICS, DFHEJVR. (DFHEJVS, DFHEJCF y DFHEJVR no se incluyen en la lista predeterminada de grupos de inicio de CICS, DFHLIST).

Nota: En la mayoría de los casos, los valores de los atributos RECORDSIZE en las definiciones de FILE proporcionadas no deberían requerir ninguna modificación. Sin embargo, si pretende instalar más de 40 CorbaServers en el servidor lógico EJB/CORBA, consulte "Especificación del RECORDSIZE de DFHEJDIR y de DFHEJOS" en la página 410.

Para obtener información de referencia sobre las definiciones de FILE, consulte Recursos FILE.

TRANSACTION y PROGRAM

Los objetos sin estado CORBA y los enterprise beans no cuentan con definiciones de recurso PROGRAM como tal. La definición de recurso PROGRAM que es relevante para un objeto sin estado CORBA o un enterprise bean es aquel para el programa procesador de solicitudes.

Las definiciones TRANSACTION y PROGRAM predeterminadas para el receptor de peticiones proporcionado por CICS y los programas procesadores de solicitudes están en el grupo de recursos DFHIIOP, que se incluye en la lista predeterminada de grupos iniciales de CICS, DFHLIST.

Normalmente no debe ser necesario sustituir las definiciones de TRANSACTION y PROGRAM predeterminadas para el receptor de peticiones (CIRR y DFHIIRRS, respectivamente). Esta es la definición de CIRR en DFHIIOP:

```
DEFINE TRANSACTION(CIRR)      GROUP(DFHIIOP)
       PROGRAM(DFHIIRRS)      TWASIZE(0)
       PROFILE(DFHCICST)      STATUS(ENABLED)
       TASKDATALOC(ANY)       TASKDATAKEY(USER)
       RUNAWAY(SYSTEM)        SHUTDOWN(ENABLED)
       PRIORITY(1)            TRANCLASS(DFHTCL00)
       DTIMOUT(NO)            TPURGE(NO)
       SPURGE(YES)            ISOLATE(NO)
       RESSEC(NO)             CMDSEC(NO)
       RESTART(NO)
       DESCRIPTION(Transacción del receptor de peticiones IIOP de CICS predeterminado)
```

Una razón para crear sus propias definiciones de TRANSACTION y PROGRAM para el programa procesador de solicitudes es especificar un perfil de JVM distinto al predeterminado. El nombre del perfil de JVM que utilizar se especifica en la opción JVMPROFILE de la definición PROGRAM para el programa procesador de solicitudes. La definición de PROGRAM predeterminada para el procesador de solicitudes (DFJIIRP en DFHIIOP) especifica el perfil de JVM DFHJVMCD. Esta es la definición de DFJIIRP en DFHIIOP:

```
DEFINE PROGRAM(DFJIIRP)       GROUP(DFHIIOP)
     DESCRIPTION(Procesador de solicitudes IIOP de CICS)
     JVM(YES)
     JVMCLASS(com.ibm.cics.iiop.RequestProcessor)
     JVMPROFILE(DFHJVMCD)
     LANGUAGE(LE370)
     RELOAD(NO)
     EXECKEY(USER)
     RESIDENT(NO)
     USAGE(NORMAL)
     USELPACOPY(NO)
     STATUS(ENABLED)
     CEDF(NO)
     DATALOCATION(ANY)
     DYNAMIC(NO)
```

Nota: El atributo CEDF se puede definir como YES para fines de depuración. Consulte “Utilización de EDF con enterprise beans” en la página 320.

Si finalmente crea su propia definición PROGRAM para el procesador de solicitudes, puede proporcionar una con cualquier nombre, pero el parámetro JVMCLASS se debe definir como **com.ibm.cics.iiop.RequestProcessor**. Elija otro perfil de JVM para que lo utilice el procesador de solicitudes, y especifique el nombre de su perfil de JVM en la opción JVMPROFILE. CICS proporciona perfiles de JVM de ejemplo en el directorio `/usr/lpp/cicsts/cicsts42/JVMProfiles` de z/OS UNIX, donde `/usr/lpp/cicsts/cicsts42` es el directorio de instalación para archivos de CICS en z/OS UNIX. En “Configuración de JVM en agrupación” en la página 95 se indica cómo ubicar, elegir y personalizar perfiles de JVM.

TCPIPSERVICE

Proporcione e instale definiciones de recurso TCPIPSERVICE para configurar la región de escucha CICS Listener para que reciba solicitudes IIOP y llame al *receptor de peticiones* IIOP . La definición de recurso TCPIPSERVICE también especifica opciones de equilibrio de carga y de seguridad. Consulte “Configuración de TCP/IP para IIOP” en la página 407.

CICS proporciona, en el grupo de recursos DFH$EJB, una definición TCPIPSERVICE para utilizar con el programa de verificación de la instalación (IVP) de EJB y la aplicación de ejemplo "Hello World" de EJB. Si se va a configurar un servidor EJB de CICS, le sugerimos que siga el ejemplo paso a paso de cómo configurar esta definición que aparece en "Acciones necesarias en CICS" en la página 260.

CORBASERVER

Proporcione e instale una definición de recurso CORBASERVER. Tenga en cuenta que es necesario definir, instalar y hacer disponible el archivo DFHEJDIR antes de poder instalar un CORBASERVER.

CICS proporciona, en el grupo de recursos DFH$EJB, una definición CORBASERVER para utilizar con el programa IVP de EJB y la aplicación de ejemplo "Hello World" de EJB. Si se va a configurar un servidor EJB de CICS, le sugerimos que siga el ejemplo paso a paso de cómo configurar esta definición que aparece en "Acciones necesarias en CICS" en la página 260.

REQUESTMODEL

Proporcione e instale definiciones de recurso REQUESTMODEL para permitir que el *receptor de peticiones* coincida con la solicitud entrante para una transacción de CICS, para definir parámetros de ejecución que se utilicen si se crea una nueva instancia del procesador de solicitudes para gestionar la solicitud. La definición de TRANSID en REQUESTMODEL predeterminada es CIRP, que especifica el programa procesador de solicitudes predeterminado DFJIIRP. Si opta por utilizar su propia definición TRANSACTION, debe definirla e instalarla; esta debe especificar una definición PROGRAM con el parámetro JVMCLASS definido como **com.ibm.cics.iiop.RequestProcessor**. Consulte "Obtención de un TRANSID de CICS" en la página 418.

Nota:

1. Debe proporcionar definiciones REQUESTMODEL únicamente si el TRANSID predeterminado, CIRP, es inadecuado o si quiere segregar la carga de trabajo de IIOP por ID de transacción (por ejemplo, para supervisión).
2. La definición TRANSACTION para CIRP especifica DYNAMIC(NO). Si desea utilizar direccionamiento dinámico de solicitudes de método para enterprise beans y objetos sin estado CORBA, debe proporcionar una o más definiciones TRANSACTION que especifiquen DYNAMIC(YES), y especificarlas en sus definiciones REQUESTMODEL.
3. Una vez que el CorbaServer esté operativo, puede utilizar la transacción CREA provista por CICS para visualizar los ID de transacción asociados con beans y métodos de bean concretos en el CorbaServer. Puede cambiar los ID de transacción, aplicar los cambios y guardar dichos cambios en las nuevas definiciones REQUESTMODEL. Este es un método más fácil que compilar definiciones REQUESTMODEL a mano.
4. En un servidor lógico de CICS multirregión, se recomienda instalar las definiciones REQUESTMODEL en las AOR así como en las regiones de escucha; consulte Figura 38 en la página 414. Las definiciones REQUESTMODEL de las AOR son necesarias para las solicitudes salientes a objetos locales. Si un objeto CORBA sin estado o un enterprise hace una llamada a otro objeto y dicho objeto está disponible en la AOR local, CICS no envía la solicitud a una región de escucha. En su lugar, ejecuta el método llamado en la tarea actual ("bucle de retorno hermético") o inicia otro procesador de solicitudes en la AOR local ("bucle de retorno normal"). Si se utiliza el bucle de retorno normal, es preferible que la nueva tarea del

procesador de solicitudes utilice el mismo REQUESTMODEL que el utilizado para llamar al primer objeto, o pueden producirse resultados imprevisibles. Si los objetos CORBA sin estado y los enterprise beans no realizan llamadas salientes, las REQUESTMODEL de la AOR no son estrictamente necesarias.

DJAR
Proporcione e instale definiciones de recurso DJAR para todos los enterprise beans.

Nota: Las definiciones de DJAR generalmente las crea y las instala el mecanismo de exploración de CICS durante el despliegue (consulte Recursos DJAR en la Guía de definición de recursos).

La Figura 38 muestra las definiciones de RDO necesarias para definir un servidor lógico de CICS. Muestra qué definiciones son necesarias en las regiones de escucha, cuáles en las AOR y cuáles en ambas.

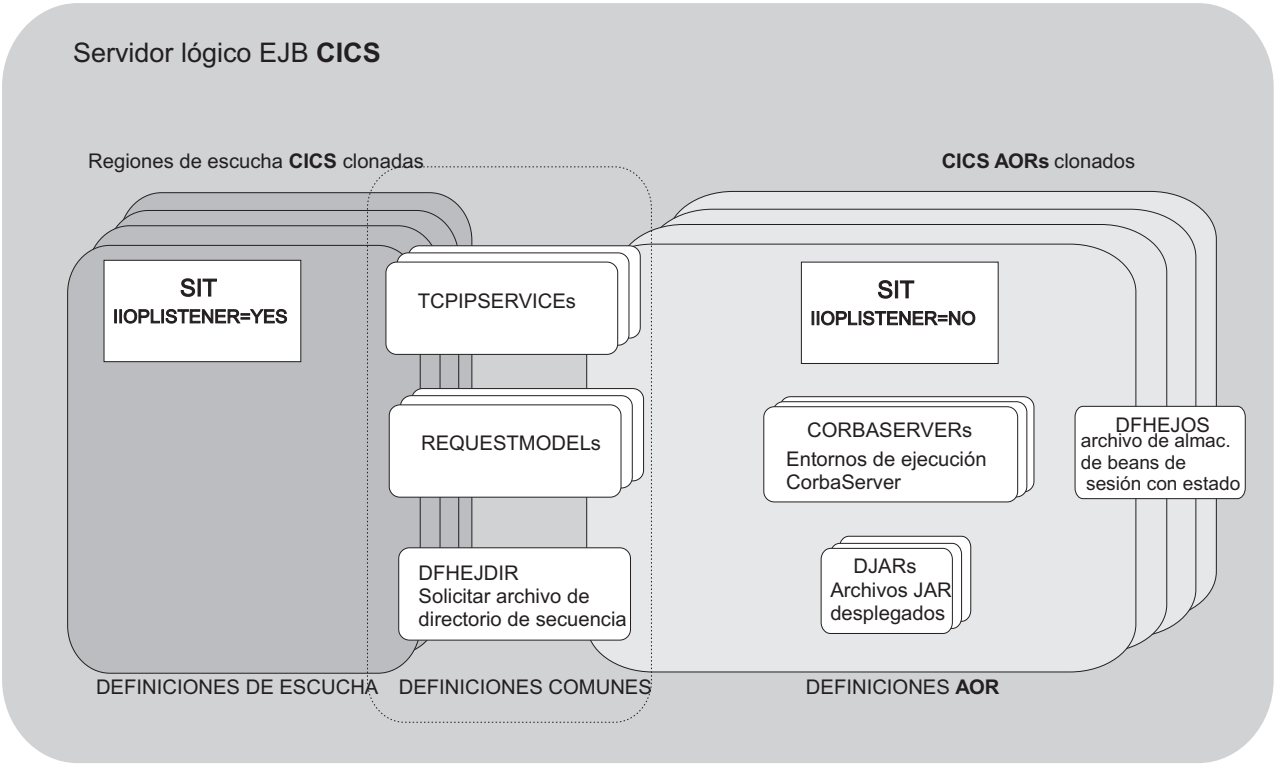


Figura 38. Definiciones de recurso en un servidor lógico de CICS. La imagen muestra qué definiciones son necesarias en las regiones de escucha, cuáles en las AOR y cuáles en ambas.

Proceso de solicitudes IIOP

El receptor de peticiones de CICS deriva un USERID y un TRANSID de CICS que establecen parámetros de ejecución de CICS para la solicitud, antes de pasar el control al procesador de solicitudes de IIOP para que invoque los métodos del destino.

Obtención de un ID de usuario de CICS

Para las solicitudes de IIOP, puede autenticar e identificar al usuario de los siguientes modos.

Acerca de esta tarea

1. Mediante autenticación de cliente de capa de sockets seguros (SSL). Consulte la *Guía de seguridad RACF de CICS* para obtener más información.
2. Si la autenticación SSL no proporciona un ID de usuario, puede utilizar el programa de seguridad sustituible por el usuario de IIOP para proporcionar uno. Especifique el nombre del programa de seguridad de IIOP en el atributo URM de la definición TCPIPSERVICE para el puerto. Consulte "Utilización del programa de seguridad IIOP sustituible por el usuario" en la página 417 para obtener más información.
3. Si ninguno de estos mecanismos proporciona un ID de usuario, se utiliza el ID de usuario predeterminado.

Si se especifica el nombre de un programa de seguridad en la definición de TCPIPSERVICE pero se omite la definición de recurso PROGRAM para él, CICS intenta compilar una definición de recurso (instalación automática) para él; si esto no funciona, o si el programa de seguridad no devuelve ningún ID de usuario, utiliza el ID de usuario asociado con el certificado de cliente SSL, si hay alguno. En caso contrario se utiliza el ID de usuario predeterminado.

La siguiente área de comunicaciones se pasa al programa sustituible por el usuario. Esta estructura se basa en el formato de un mensaje de IIOP definido en *The Common Object Request Broker: Architecture and Specification*, que se puede obtener en el sitio web de OMG en http://www.omg.org/library

Desplazamiento hex.	Tipo	Longitud	Nombre
(0)	STRUCTURE	80	sXOPUS
(0)	CHARACTER	4	standard_header
(4)	FULLWORD	4	pIIOPData
(8)	FULLWORD	4	lIIOPData
(C)	FULLWORD	4	pRequestBody
(10)	FULLWORD	4	lRequestBody
(14)	CHARACTER	4	corbaserver
(18)	FULLWORD	4	pBeanName
(1C)	FULLWORD	4	lBeanName
(20)	FULLWORD	4	BeanInterfaceType
(24)	FULLWORD	4	pModule
(28)	FULLWORD	4	lModule
(2C)	FULLWORD	4	pInterface
(30)	FULLWORD	4	lInterface

Desplazamiento hex.	Tipo	Longitud	Nombre
(34)	FULLWORD	4	pOperation
(38)	FULLWORD	4	lOperation
(3C)	CHARACTER	8	userid
(44)	FULLWORD	4	transid
(48)	FULLWORD	4	flag_bytes
(4C)	FULLWORD	4	return_code
(50)	FULLWORD	4	reason_code

standard_header
Contiene una cabecera estándar con el siguiente formato:

function
Campo de 1 byte definido como X'00'

domain
Campo de 2 caracteres que contiene II

* Campo reservado de 1 carácter

pIIOPData
Contiene la dirección del primer megabyte del almacenamiento intermedio de IIOP sin convertir.

lIIOPData
Contiene la longitud del almacenamiento intermedio de IIOP sin convertir.

pRequestbody
Contiene la dirección de la solicitud de IIOP entrante.

lRequestbody
Contiene la longitud de la solicitud de IIOP entrante.

corbaserver
Contiene el nombre del CorbaServer asociado con esta solicitud.

pBeanName
Contiene un puntero para el nombre de bean de EBCDIC.

lBeanName
Contiene la longitud del nombre de bean.

BeanInterfaceType
Contiene un valor enumerado. X'00' indica local, X'01' indica remoto.

pModule
Contiene un puntero para el nombre de módulo de EBCDIC.

lModule
Contiene la longitud del nombre de módulo.

pInterface
Contiene un puntero para el nombre de interfaz de EBCDIC.

lInterface
Contiene la longitud del nombre de interfaz.

pOperation
Contiene un puntero para el nombre de operación de EBCDIC.

lOperation
Contiene la longitud de la operación.

userid
Contiene el ID de usuario de entrada y de salida. El ID de usuario de salida debe tener exactamente 8 caracteres. Si tiene menos de 8 caracteres, debe rellenarse con espacios en blanco.

transid
Contiene el TRANSID de entrada.

Flag_bytes
Contiene los siguientes indicadores:

littleEndian
Campo de 1 byte que muestra el orden de bytes, donde **1** indica VERDADERO y **0** indica FALSO

sslClientUserid
Campo de 1 byte que muestra la derivación del ID de usuario si se especifica SSLTYPE CLIENTAUTH en la definición de TCPIPSERVICE, donde:

0 Identificador de usuario definido desde DFLTUSER

1 Identificador de usuario definido desde SSL CERTIFICATE

* Campo reservado de 2 bytes.

return_code
Contiene el código de retorno.

reason_code
Contiene el código de razón.

RETNCODE se define como RCUSRID (X'01') si se va a devolver un ID de usuario. Un programa sustituible por el usuario debe devolver todos los demás campos sin modificar o se producirán resultados imprevisibles.

Para obtener información acerca de la instalación de programas sustituibles por el usuario, consulte Customizing with user-replaceable programs en la *Guía de personalización de CICS*.

Utilización del programa de seguridad IIOP sustituible por el usuario:

Como opción, puede proporcionar un programa de seguridad IIOP para examinar elementos de la solicitud IIOP entrante y generar un USERID.

Debe especificar el nombre del programa de seguridad en el atributo URM de la definición de recurso TCPIPSERVICE y también proporcionar una definición de recurso PROGRAM para él. Si no especifica un valor para URM en TCPIPSERVICE, no se llama a ningún programa.

El programa de seguridad IIOP solo se llama si CICS no puede obtener un ID de usuario mediante autenticación de cliente SSL. Consulte SSL authentication, en la *Guía de seguridad RACF de CICS*, para obtener más información.

Se proporciona un programa de seguridad IIOP de ejemplo, DFHXOPUS.

El programa de seguridad puede utilizar servicios de CICS, como una salida de usuario relacionada con tareas, para acceder a DB2, y parámetros de aplicación codificados en el cuerpo de la solicitud.

Utilización de DFHXOPUS:

El programa de ejemplo sustituible por el usuario proporcionado por CICS, DFHXOPUS, acepta el ID de usuario de RACF asociado con el certificado de cliente, si hay alguno.

Si no hay ningún ID de usuario de RACF asociado con un certificado:

- Para SSL(CLIENTAUTH), DFHXOPUS utiliza los primeros ocho caracteres del COMMONNAME extraído del certificado de cliente.
- Para SSL(YES) o SSL(NO), DFHXOPUS utiliza los primeros ocho caracteres del principal de IIOP, si hay alguno.

  Nota: Las versiones del protocolo Inter-ORB general (GIOP) de la 1.2 en adelante no soportan el campo del principal de IIOP en las cabeceras de solicitud. Por lo que DFHXOPUS solo devolverá un ID de usuario derivado del principal de IIOP si la solicitud está en formato GIOP 1.1 o anterior.

Si no se ha encontrado ningún ID de usuario mediante estos procedimientos, DFHXOPUS devuelve el ID de usuario especificado en el parámetro de inicialización del sistema CICS. Parámetro de inicialización del sistema DFLTUSERDFLTUSER.

El programa de salida de seguridad devuelve el ID de usuario en el campo `userid` del área de comunicaciones. Si el ID de usuario tiene menos de 8 caracteres, el programa de salida rellena el campo con espacios en blanco. Como se devuelve un ID de usuario, el campo `return_code` se define como RCUSRID (X'01') .

Si escribe su propio programa de salida de seguridad, debe devolver todos los campos menos `userid` y `return_code` sin modificar, o podrían producirse resultados imprevisibles.

Obtención de un TRANSID de CICS

Para asociar la solicitud de GIOP entrante con un ID de transacción de CICS, es necesario proporcionar la definición de recurso REQUESTMODEL.

Debe proporcionar recursos REQUESTMODEL para todas las solicitudes posibles que deban ejecutarse en un ID de transacción que no sea el predeterminado. En el tiempo de ejecución, cuando CICS recibe una solicitud GIOP compara campos de la solicitud con valores predefinidos en los REQUESTMODEL, para encontrar el REQUESTMODEL que coincida más exactamente con la solicitud. El REQUESTMODEL seleccionado proporciona el nombre del TRANSID que se utiliza para procesar la solicitud. Si no se encuentra ninguna coincidencia se utiliza un TRANSID predeterminado (CIRP). Los REQUESTMODEL se pueden utilizar con enterprise beans, objetos CORBA sin estado o ambos. Especifican:

- Patrones CORBA de MODULE e INTERFACE con los que comparar solicitudes de objetos CORBA sin estado.
- Nombres de bean para búsqueda de coincidencia de enterprise beans.
- Patrones de OPERATION con los que comparar:
  - Nombres de método de enterprise bean.

– Nombres de método de objeto CORBA sin estado.
– Operaciones de IDL (solo objetos CORBA sin estado).

Nota: El campo OPERATION está sujeto a las reglas de alteración de nombres de Java a IDL descritas en "Alteración de nombres del campo OPERATION" en la página 420.

- La transacción de CICS que lanzar cuando se recibe una solicitud coincidente. El valor predeterminado es CIRP, que especifica el programa predeterminado DFJIIRP. Si opta por utilizar su propia definición de transacción, debe basarla en CIRP y proporcionar una definición de recurso TRANSACTION con el parámetro PROGRAM definido como el nombre del programa de CICS que está definido con el parámetro JVMCLASS definido como **com.ibm.cics.iiop.RequestProcessor**. CICS proporciona las siguientes definiciones de recurso predeterminadas en el grupo DFHIIOP:

```
DEFINE TRANSACTION(CIRP)    GROUP(DFHIIOP)
     PROGRAM(DFJIIRP)       TWASIZE(0)
     PROFILE(DFHCICST)      STATUS(ENABLED)
     TASKDATALOC(ANY)       TASKDATAKEY(USER)
     RUNAWAY(SYSTEM)        SHUTDOWN(ENABLED)
     PRIORITY(1)            TRANCLASS(DFHTCL00)
     DTIMOUT(NO)            TPURGE(NO)
     SPURGE(YES)            ISOLATE(YES)
     RESSEC(YES)            CMDSEC(YES)
     RESTART(NO)
     DESCRIPTION(Transacción del procesador de solicitudes IIOP de CICS predeterminado)


DEFINE PROGRAM(DFJIIRP)       GROUP(DFHIIOP)
     DESCRIPTION(Procesador de solicitudes IIOP de CICS)
     JVM(YES)
     JVMCLASS(com.ibm.cics.iiop.RequestProcessor)
     JVMPROFILE(DFHJVMCD)
     LANGUAGE(LE370)        RELOAD(NO)             EXECKEY(USER)
     RESIDENT(NO)           USAGE(NORMAL)          USELPACOPY(NO)
     STATUS(ENABLED)        CEDF(NO)               DATALOCATION(ANY)
     DYNAMIC(NO)
```

  Consulte "Direccionamiento dinámico" en la página 421 si la solicitud se debe dirigir a un AOR.
- El nombre del CorbaServer que procesará la solicitud.

Consulte la *Guía de definición de recurso de CICS* para obtener detalles completos de la definición de recurso REQUESTMODEL.

Nota: Para simplificar el proceso de creación de definiciones de REQUESTMODEL para enterprise beans, utilice la transacción CREA proporcionada por CICS.

Coincidencia de patrones:

Todas las solicitudes se comparan con los valores instalados de REQUESTMODEL para CORBASERVER y TYPE.

Un valor TYPE de CORBA indica la solicitud de un objeto CORBA sin estado, un valor TYPE de EJB indica una solicitud de un enterprise bean, mientras que un valor TYPE de GENERIC puede indicar cualquier tipo de solicitud. Después se buscan más coincidencias en función del valor de TYPE:

Objetos CORBA sin estado

Para objetos CORBA sin estado (TYPE=CORBA o GENERIC), el proceso de coincidencia compara el nombre **MODULE** y los campos **INTERFACE** y **OPERATION** contenidos en el mensaje de IIOP con los patrones definidos en cada REQUESTMODEL instalado, hasta que se encuentra la coincidencia más cercana. INTERFACE, MODULE y OPERATION se pueden definir como patrones genéricos. La reglas para búsqueda de coincidencia de patrón se resumen de la siguiente forma:

- Como separadores de componentes se utilizan los dos puntos dobles. Cada componente debe tener entre 1 y 16 caracteres.
- Los patrones genéricos pueden constar de cero o más caracteres seguidos de un *.

Si varios patrones genéricos coinciden con una cadena determinada, el patrón genérico más largo es la coincidencia más específica.

Enterprise beans
Para enterprise beans, el proceso de búsqueda de coincidencia compara los campos BEANNAME, OPERATION e INTFACETYPE del mensaje de IIOP con los definidos en cada REQUESTMODEL instalado.

Alteración de nombres del campo OPERATION:

El campo OPERATION de la definición REQUESTMODEL se utiliza para proporcionar el nombre del método remoto con el que este modelo de solicitud va a hacerse coincidir.

La solicitud GIOP recibida en el tiempo de ejecución incluye un campo de operación que se compara con el campo OPERATION del modelo de solicitud. Sin embargo, el valor del campo de operación no siempre es igual al nombre del método, como se utiliza en el objeto CORBA sin estado o en el enterprise bean. Si se utiliza RMI-IIOP (como sucede siempre en el caso de los enterprise beans y puede suceder con los objetos CORBA sin estado), el nombre del método pasa por un proceso conocido como "*alteración*" para cambiar el nombre del método a una forma canónica adecuada para transmisión mediante IIOP. Este nombre de método alterado puede no ser el mismo que el nombre de método original. El campo de operación en REQUESTMODEL debe proporcionar la versión alterada del nombre del método (o un patrón, mediante caracteres comodín, que coincida con él).

La transacción CREA que proporciona CICS se puede utilizar para crear definiciones REQUESTMODEL para enterprise beans que gestionen automáticamente este problema de alteración de nombres.

Este conocimiento de alteración y reconversión de nombres se compila en las clases stub (apéndice) y tie (enlace) de la aplicación, generadas mediante el compilador RMI (RMIC).

Para obtener más información sobre la alteración, consulte "Alteración de nombres para Java" en la página 421.

Ejemplos de REQUESTMODEL:

Este es un ejemplo de un objeto CORBA sin estado REQUESTMODEL:

```
DEFINE REQUESTMODEL(DFJ$IIRH)  GROUP(DFH$IIOP)
       CORBASERVER(IIOP)
       TYPE(Corba)
       MODULE(hello)
       INTERFACE(HelloWorld)
```

```
OPERATION(*)
TRANSID(IIHE)
DESCRIPTION(ejemplo de servidor Java Hello world)
```

Direccionamiento dinámico:

Si la invocación de método se va a dirigir a otra región (AOR), debe definir el TRANSID especificado en REQUESTMODEL como que se puede dirigir de forma dinámica en la región de escucha (utilizando el parámetro DYNAMIC). Si utiliza la definición TRANSACTION predeterminada, CIRP, tendrá que cambiarlo.

Alteración de nombres para Java

La alteración de nombres es un término que denota el proceso de correlación de un nombre que sea válido en un lenguaje de programación concreto con un nombre que sea válido en el Interface Definition Language (IDL) de CORBA. Esta sección explica por qué es necesaria la alteración para nombres Java, cómo se alteran los nombres y cómo afecta la alteración a su sistema CICS.

Por qué es necesaria la alteración para los nombres de Java:

Los programas cliente de Java utilizan la invocación a método remoto (RMI) de Java para invocar métodos en un servidor.

A su vez, la RMI utiliza uno de estos dos protocolos de comunicación entre cliente y servidor:

Java Remote Method Protocol (JRMP)
RMI utiliza el JRMP cuanto tanto la aplicación cliente como la aplicación de servidor están escritas en Java. CICS no utiliza JRMP.

Protocolo Internet Inter-ORB (IIOP)
RMI lo utiliza en un entorno en el que la aplicación cliente y la aplicación de servidor puedan estar escritas en idiomas distintos. Cuando se utiliza IIOP como protocolo de comunicación, las aplicaciones cliente de Java pueden utilizar el RMI para invocar programas de servidor en otro lenguaje (por ejemplo, C++), así como para invocar programas Java remotos.

IIOP utiliza Interface Definition Language (IDL) para especificar interfaces entre objetos de una forma independiente del lenguaje. Cuando un cliente Java realiza una llamada a método remoto, el nombre del método Java y sus argumentos se convierten en el IDL equivalente para transmisión al servidor utilizando IIOP. En este punto la alteración puede ser necesaria, porque hay muchas diferencias en las reglas para nombres Java y nombres IDL. Algunas de estas diferencias son las siguientes:

- Los nombres Java diferencian entre mayúsculas y minúsculas, los nombres IDL no.
- Java soporta métodos sobrecargados, IDL no.
- Los nombres Java pueden contener caracteres Unicode, los nombres IDL no pueden.
- Algunos nombres Java válidos pueden chocar con palabras clave de IDL.
- Los nombres Java pueden empezar por un subrayado inicial, los nombres IDL no pueden.

En estos casos, así como en otros, los nombres Java que no están permitidos en IDL, o que están permitidos pero pueden ser ambiguos, se alteran para darles una forma aceptable.

Cómo se alteran los nombres de Java:

Las reglas por las que una llamada a método de Java se correlaciona con un nombre de IDL no son sencillas y dependen de las circunstancias.

Aquí tiene un ejemplo:

> Una interfaz remota de Java tiene los métodos save, Save y SAVE. Estos nombres se diferencian claramente en Java, pero -debido a que los nombres de IDL no distinguen entre mayúsculas y minúsculas- IDL no puede diferenciarlos. Por lo tanto, los nombres se alteran para que se distingan. Los nombres alterados son save_, Save_0 y SAVE_0_1_2_3. Sin embargo, si la interfaz remota de Java tuviera un único método -save-, el nombre no se alteraría, ya que no existiría la posibilidad de que se produjera ambigüedad.

Este ejemplo ilustra dos principios importantes:

- No es posible determinar el nombre alterado de un método determinado sin saber qué otros métodos hay.
- Añadir o eliminar un método puede afectar a los nombres alterados de otros métodos.

Otros casos en los que la alteración puede ser necesaria se manejan de forma distinta. Para obtener información detallada sobre la correlación entre Java e IDL, consulte *Java Language to IDL Mapping*, publicado por el grupo de gestión de objetos (OMG) (`http://www.omg.org`).

Cómo afecta la alteración a CICS:

Aunque el soporte para IIOP en CICS contiene código que implementa las reglas de alteración, hay muy poco efecto visible en la forma de configurar y utilizar el sistema CICS.

Solo hay dos situaciones en las que debe saber que se produce alteración. Estas son:

Al definir REQUESTMODEL.
Las definiciones de recursos REQUESTMODEL correlacionan solicitudes IIOP de entrada con transacciones de CICS. Cuando se recibe una solicitud de entrada lanzada por una invocación de método remoto Java, el atributo OPERATION de REQUESTMODEL se compara con el nombre alterado que hay en la solicitud de entrada para determinar si REQUESTMODEL coincide con la solicitud. Si es posible que se produzca alteración, no especifique ningún nombre de método en el atributo OPERATION de REQUESTMODEL, y en su lugar especifique una operación genérica.

Al crear perfiles de depuración para programas Java.
Los perfiles de depuración especifican qué instancias de programa se van a ejecutar bajo el control de un depurador. Cuando se recibe una solicitud de entrada lanzada por una invocación de método remoto Java, el campo de método del perfil de depuración se compara con el nombre alterado que hay en la solicitud de entrada para determinar si el perfil coincide con la

solicitud. Si es posible que se produzca alteración, no especifique ningún nombre de método en el perfil de depuración y, en su lugar, especifique un método genérico.

PRECAUCIÓN: Aunque -en teoría- resulta posible deducir los nombres alterados correspondientes a cada método, no es una tarea sencilla y no lo aconsejamos. Para ello, necesitará un gran conocimiento de las reglas de alteración y de todos los nombres de método utilizados en su aplicación. También existe el riesgo de que pequeños cambios en una aplicación puedan modificar el nombre alterado.

Manejo de diagnóstico de IIOP

Si un método remoto que se invoca mediante IIOP sufre una anomalía, el código de cliente recibirá una excepción de CORBA. Esto incluye todas las excepciones de enterprise bean.

Las excepciones de CORBA se definen en la documentación de CORBA, que se puede obtener en el sitio web de CORBA: `http://www.omg.org`.

En muchos casos, la excepción incluye un código menor específico de CICS para ayudar en la determinación de problemas. Actualmente CICS utiliza los siguientes códigos menores:

Tabla 24. Códigos menores de CORBA específicos de CICS

Código	**Componente de CICS que detecta el problema**
1229111296	Receptor de peticiones de IIOP de CICS.
1229111297	Cualquier otro lugar del dominio de CICS II
1229111298	Componente ORB del dominio OT de CICS
1229111299	Componente JTS del dominio OT de CICS
1229111300	Componente CSI del dominio OT de CICS
1229111301	Componente CSI del dominio EJ de CICS

Si el cliente recibe una excepción CORBA que contenga cualquiera de los códigos menores de CICS, debería examinar los registros de mensajes de CICS para obtener más información sobre el error.

Avisos

Esta información se desarrolló para los productos y servicios ofrecidos en los EE. UU. Es posible que IBM no ofrezca en otros países los productos, servicios o características que se explican en este documento. Consulte con su representante de IBM local para obtener información sobre los productos y servicios disponibles en su zona actualmente. Las referencias a productos, programas o servicios IBM no pretenden afirmar ni implicar que sólo pueda utilizarse ese producto, programa o servicio IBM. En su lugar, puede utilizarse cualquier producto, programa o servicio equivalente que no vulnere ningún derecho de propiedad intelectual de IBM. Sin embargo, es responsabilidad del usuario evaluar y verificar la operación de cualquier producto, programa o servicio que no sea de IBM.

IBM puede tener patentes o solicitudes de patentes pendientes que abarquen el tema descrito en este documento. La provisión de este documento no le otorga ninguna licencia para estas patentes. Puede enviar consultas sobre licencias, por escrito, a:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
EE. UU.

Si tiene preguntas sobre licencia referentes a información de doble-byte (DBCS), póngase en contacto con el Departamento de Propiedad Intelectual de IBM de su país, o envíe sus consultas, por escrito, a:

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

El siguiente párrafo no es válido para el Reino Unido ni cualquier otro país donde estas disposiciones no sean consistentes con la legislación local:

INTERNATIONAL BUSINESS MACHINES CORPORATION OFRECE ESTA PUBLICACIÓN "TAL COMO SE PRESENTA" SIN NINGUNA GARANTÍA DE NINGÚN TIPO, YA SEA EXPRESA O IMPLÍCITA, INCLUYENDO, PERO SIN LIMITARSE A ELLAS, LAS GARANTÍAS IMPLÍCITAS DE NO INFRACCIÓN, COMERCIABILIDAD O ADECUACIÓN PARA UN PROPÓSITO CONCRETO. Algunos estados no permiten ninguna declaración de limitación de responsabilidad de garantías expresas o implícitas en ciertas transacciones, por lo que esta declaración puede no ser válida para usted.

Esta publicación podría incluir inexactitudes técnicas o errores tipográficos. La información que aparece aquí se somete a cambios periódicos; estos cambios se incorporarán en nuevas ediciones de la publicación. IBM puede introducir mejoras y/o cambios al producto o productos y/o al programa o los programas descritos en esta publicación en cualquier momento y sin previo aviso.

Los poseedores de licencias de este programa que deseen tener información sobre él con el propósito de permitir: (i) el intercambio de información entre programas

creados independientemente y otros programas (incluido este) y (ii) la utilización mutua de la información que se ha intercambiado, deberían contactar con IBM United Kingdom Laboratories, MP151, Hursley Park, Winchester, Hampshire, Inglaterra, SO21 2JN.: Dicha información puede estar disponible sujeta a los términos y condiciones adecuados, lo que en algunos casos incluirá el pago de una tarifa.

El programa bajo licencia descrito en esta información y todo el material con licencia disponible para el mismo los proporciona IBM bajo los términos del Acuerdo de cliente IBM, el Acuerdo de licencia de programa internacional de IBM o cualquier acuerdo equivalente entre nosotros.

Marcas registradas

IBM, el logotipo de IBM, e ibm.com son marcas registradas de International Business Machines Corp registradas en varias jurisdicciones de distintas partes del mundo. Otros nombres de servicios o productos pueden ser marcas registradas de IBM o de otras empresas. Hay disponible una lista actual de marcas registradas de IBM en la web, en "Copyright and trademark information" (Información sobre copyright y marcas registradas), en www.ibm.com/legal/copytrade.shtml.

Java y todas las marcas registradas o logotipos basados en Java son marcas registradas de Oracle y de sus filiales.

Linux es una marca registrada de Linus Torvalds en los Estados Unidos, otros países o ambos.

Microsoft y Windows son marcas registradas de Microsoft Corporation en los Estados Unidos, otros países o ambos.

UNIX es una marca registrada de The Open Group en los Estados Unidos y otros países.

Otros nombres de servicios o productos pueden ser marcas registradas de IBM o de otras empresas.

Bibliografía

Libros de CICS para CICS Transaction Server para z/OS

Generales

CICS Transaction Server para z/OSDirectorio de programas, GI13-0565
CICS Transaction Server para z/OSNovedades, GC34-7192
CICS Transaction Server para z/OSActualización desde CICS TS versión 3.1, GC11-7905
CICS Transaction Server para z/OS Actualización desde CICS TS versión 3.2, GC11-7962
CICS Transaction Server para z/OSActualización desde CICS TS versión 4.1, GC11-7963
CICS Transaction Server para z/OS Guía de instalación, GC34-7171

Acceso a CICS

Guía de acceso a Internet de CICS, SC34-7173
Guía de servicios web de CICS, SC34-7191

Administración

Guía de definición del sistema CICS, SC34-7185
Guía de personalización de CICS, SC34-7161
Guía de definición de recurso de CICS, SC34-7181
Guía de operaciones y programas de utilidad de CICS, SC34-7213
Guía de seguridad RACF de CICS, SC34-7179
Transacciones suministradas de CICS, SC34-7184

Programación

Guía de programación de la aplicación de CICS, SC34-7158
Referencia de programación de la aplicación de CICS, SC34-7159
Referencia de programación del sistema CICS, SC34-7186
Guía del usuario de la interfaz de programación de aplicaciones para usuarios, SC34-7169
Bibliotecas de clases C++ OO de CICS, SC34-7162
Guía de programación de transacción distribuida de CICS, SC34-7167
CICS Business Transaction Services, SC34-7160
Aplicaciones Java en CICS, SC34-7174

Diagnóstico

Guía de determinación de problemas de CICS, GC34-7178
Guía de rendimiento de CICS, SC34-7177
Mensajes y códigos de CICS Vol 1, GC34-7175
Mensajes y códigos de CICS Vol 2, GC34-7176
Referencia de diagnóstico de CICS, GC34-7166
Guía de recuperación y reinicio de CICS, SC34-7180
Áreas de datos de CICS, GC34-7163
Entradas de rastreo de CICS, SC34-7187
Referencia de interfaces de herramientas de depuración de CICS, GC34-7165

Comunicación

Guía de intercomunicación de CICS, SC34-7172
Guía de interfaces externas de CICS, SC34-7168

Bases de datos

Guía de DB2 de CICS, SC34-7164

Guía de control de bases de datos IMS de CICS, SC34-7170

Guía de tablas de datos compartidos de CICS, SC34-7182

Libros de CICSPlex SM para CICS Transaction Server para z/OS

Generales

Conceptos y planificación de CICSPlex SM, SC34-7196
Guía de interfaz de usuario web CICSPlex SM, SC34-7214

Administración y gestión

Administración CICSPlex SM, SC34-7193
Referencia de vistas de operaciones de CICSPlex SM, SC34-7202
Referencia de vistas de supervisión de CICSPlex SM, SC34-7200
Cargas de trabajo de gestión de CICSPlex SM, SC34-7199
Uso de recursos de gestión de CICSPlex SM, SC34-7198
Aplicaciones de negocio de gestión de CICSPlex SM, SC34-7197

Programación

Guía de programación de la aplicación de CICSPlex SM, SC34-7194
Referencia de programación de la aplicación de CICSPlex SM, SC34-7195

Diagnóstico

Referencia de tablas de recursos de CICSPlex SM Vol 1, SC34-7204
Referencia de tablas de recursos de CICSPlex SM Vol 2, SC34-7205
Mensajes y códigos de CICSPlex SM, GC34-7201
Determinación de problemas de CICSPlex SM, GC34-7203

Otras publicaciones sobre CICS

En las siguientes publicaciones hay más información acerca de CICS, pero no se incluyen como parte de CICS Transaction Server para z/OS, Versión 4 Release 2.

Diseño y programación de aplicaciones de CICS, SR23-9692

Guía de ayuda para la migración de aplicaciones de CICS, SC33-0768

Familia CICS: estructura de la API, SC33-1007

Familia CICS: programación cliente/servidor, SC33-1435

Familia CICS: comunicación interproducto, SC34-6853

Familia CICS: comunicación desde CICS en el sistema/390, SC34-6854

Pasarela de transacción de CICS para administración de z/OS, SC34-5528

Familia CICS: información general, GC33-0155

Guía de aplicaciones de muestra de CICS 4.1, SC33-1173

Guía XRF de CICS/ESA 3.3 , SC33-0661

Otras publicaciones de IBM

Las siguientes publicaciones contienen información sobre productos IBM relacionados.

IBM Developer Kit and Runtime Environment, Java 2 Technology Edition Diagnostics Guide, SC34-6358

Persistent Reusable Java Virtual Machine User's Guide, SC34-6201

Accesibilidad

Las funciones de accesibilidad ayudan a los usuarios que sufren una discapacidad física, como problemas de movilidad o limitaciones en la visión, a utilizar productos de software correctamente.

Pueden realizarse la mayor parte de las tareas necesarias para configurar, ejecutar y mantener el sistema CICS de uno de estos modos:

- con el uso de un emulador 3270 con sesión iniciada en CICS
- con el uso de un emulador 3270 con sesión iniciada en TSO
- con el uso de un emulador 3270 como una consola del sistema MVS

IBM Personal Communications proporciona emulación 3270 con funciones de accesibilidad para personas con discapacidades. Puede utilizar este producto para proporcionar las funciones de accesibilidad que necesite en su sistema CICS.

Índice

Caracteres Especiales

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

Z

Hoja de Comentarios

CICS Transaction Server para z/OS
Versión 4 Release 2
Aplicaciones Java en CICS

Número de Publicación SC11-7928-01

Por favor, sírvase facilitarnos su opinión sobre esta publicación, tanto a nivel general (organización, contenido, utilidad, facilidad de lectura,...) como a nivel específico (errores u omisiones concretos). Tenga en cuenta que los comentarios que nos envíe deben estar relacionados exclusivamente con la información contenida en este manual y a la forma de presentación de ésta.

Para realizar consultas técnicas o solicitar información acerca de productos y precios, por favor diríjase a su sucursal de IBM, business partner de IBM o concesionario autorizado.

Para preguntas de tipo general, llame a "IBM Responde" (número de teléfono 901 300 000).

Al enviar comentarios a IBM, se garantiza a IBM el derecho no exclusivo de utilizar o distribuir dichos comentarios en la forma que considere apropiada sin incurrir por ello en ninguna obligación con el remitente.

Comentarios:

Gracias por su colaboración.

Para enviar sus comentarios:
- Envíelos por correo a la dirección indicada en el reverso.
- Envíelos por fax al número siguiente: +44 1962 816151
- Envíelos por correo electrónico a: idrcf@uk.ibm.com

Si desea obtener respuesta de IBM, rellene la información siguiente:

Nombre	Dirección
Compañía	
Número de teléfono	Dirección de e-mail

Hoja de Comentarios
SC11-7928-01

IBM United Kingdom Limited
User Technologies Department (MP095)
Hursley Park
Winchester
Hampshire
Reino Unido

SC11-7928-01