

CICS Transaction Server for z/OS



Uso de servicios web con CICS

Versión 5 Release 4

CICS Transaction Server for z/OS



Uso de servicios web con CICS

Versión 5 Release 4

Note

Before using this information and the product it supports, read the information in “Notices” en la página 693.

This edition applies to the IBM CICS Transaction Server for z/OS Version 5 Release 4 (product number 5655-Y04) and to all subsequent releases and modifications until otherwise indicated in new editions.

© Copyright IBM Corporation 2005, 2017.

Contenido

Acerca de este PDF. v

Capítulo 1. Servicios web y CICS 1

CICS y servicios web SOAP	4
Nodos SOAP	5
Mensajes SOAP y la estructura de datos de la aplicación	9
WSDL y la estructura de datos de la aplicación	11
WSDL y patrones de intercambio de mensajes	13
Archivo de enlace de servicio web.	15
Estándares externos.	15
Arquitectura y formato de mensaje SOAP	15
Planificación para utilizar servicios web SOAP	27
Servicios web CICS y JSON	28
Conceptos de los servicios web JSON.	29
Conceptos de servicios web JSON de RESTful	31
Planificación para utilizar servicios web JSON.	32
CICS y z/OS Connect	35

Capítulo 2. Configuración de servicios web en CICS 41

Configuración del sistema CICS para servicios web	41
Recursos de CICS para servicios web.	41
Configuración de CICS para utilizar el transporte WebSphere MQ	45
Interoperabilidad entre el asistente de servicios web y WSRR	53
Creación de la infraestructura de servicios web	54
Infraestructura de servicios web	54
Creación de la infraestructura de CICS para un proveedor de servicios SOAP	64
Creación de la infraestructura CICS para un solicitante de servicio SOAP	67
Creación de la infraestructura de CICS para un proveedor de servicios JSON	69
Creación de la infraestructura de CICS para un proveedor de servicios JSON que no es de Java	70
Configuración de z/OS Connect for CICS	71
Archivos de configuración de interconexión	81
Manejadores de aplicación	134
Manejadores de mensajes	136
Manejadores de mensajes SOAP	145
Contenedores utilizados en la interconexión	150
Proceso de tiempo de ejecución para servicios web	183
Soporte para transacciones de servicios web	193
Soporte para la optimización MTOM/XOP de datos binarios	201
Soporte para Web Services Addressing	211
Soporte para SAML	233

Capítulo 3. Desarrollo de servicios web 235

Creación de un servicio web JSON	235
--	-----

El asistente de JSON de CICS	235
Creación de una aplicación de proveedor de servicios JSON utilizando un asistente de JSON	273
Restricciones de servicio web JSON	289
Correlación y transformación de datos de aplicación y XML	291
El asistente XML de CICS	291
Generación de correlaciones desde estructuras de lenguaje	403
Generación de correlaciones desde un esquema XML	406
Transformación de datos de aplicación en XML	407
Transformación de XML en datos de aplicación	408
Correlación y transformación de datos de aplicación y JSON	410
El asistente de JSON de CICS	411
Generación de correlaciones a partir de estructuras de lenguaje	480
Generación de correlaciones a partir de un esquema JSON	481
Transformación de datos de aplicación en JSON enlazando con DFHJSON	483
Transformación de datos de aplicación en JSON utilizando el mandato de API TRANSFORM DATATOJSON	484
Transformación de JSON en datos de aplicación enlazando con DFHJSON	485
Transformación de JSON en datos de aplicación utilizando el mandato de API TRANSFORM JSONTODATA	486
Creación de una aplicación de cliente de servicio web JSON	487
Creación de un servicio web JSON	488
El asistente de servicios web de CICS	489
Creación de un proveedor de servicios web utilizando el asistente de servicios web.	576
Creación de una aplicación de proveedor de servicios a partir de una descripción de servicio web	576
Creación de una aplicación de proveedor de servicios a partir de una estructura de datos	579
Creación de un documento de descripción de canal	582
Personalización de los documentos de descripción de servicios web generados	584
Envío de un error SOAP.	585
Creación de un solicitante de servicio web utilizando el asistente de servicios web.	587
Creación de un servicio web utilizando un conjunto de herramientas	590
Creación de sus propias aplicaciones de servicio web preparadas para XML	591
Creación de una aplicación de proveedor de servicios preparada para XML.	591
Creación de una aplicación de solicitante de servicio preparada para XML	593

Uso de Java con servicios web.	595
Despliegue de un servicio web de modalidad de proveedor Java en un servidor JVM de Axis2.	595
Creación de un servicio web Java que genera y analiza XML.	597
Creación de un servicio web Java que tiene una interfaz COBOL	597
Despliegue de un servicio web JAX-WS modalidad solicitante.	598
Despliegue de un servicio web de modalidad de proveedor Java en un servidor Liberty JVM	598
Validación de mensajes SOAP	599
Manejo de datos proporcionados por la aplicación no válidos ni inicializados	600
Ejemplo 1: tolerancia de campos decimales	601
Capítulo 4. Soporte para proteger los servicios web	605
Requisitos previos para la Seguridad de servicios web	606
Planificación para proteger servicios web SOAP	606
Opciones para proteger mensajes SOAP	607
Autenticación mediante un servicio de señales de seguridad	610
Interfaz de cliente de confianza	611
Firma de mensajes SOAP	612
Algoritmos de firma	612
Ejemplo de un mensaje SOAP firmado	612
Soporte CICS para mensajes SOAP cifrados	613
Algoritmos de cifrado	614
Ejemplo de un mensaje SOAP cifrado	614
Configuración de RACF para la Seguridad de servicios web	615
Configuración de los servicios web de modalidad de proveedor para la propagación de identidad	617
Configuración de la interconexión para la Seguridad de servicios web.	620
Escritura de un manejador de seguridad personalizado	623
Invocación del cliente de confianza desde un manejador de mensajes	624
Seguridad para z/OS Connect.	625
Configuración de permisos para APIs y servicios de z/OS Connect	626

Capítulo 5. Resolución de problemas de servicios web 629

Resolución de problemas de servicios web SOAP	629
Diagnóstico de errores de despliegue	629
Diagnóstico de errores de tiempo de ejecución del proveedor de servicios	631
Diagnóstico de errores de tiempo de ejecución del solicitante de servicio	633
Diagnóstico de errores de MTOM/XOP	635
Diagnóstico de errores de conversión de datos	637
Resolución de problemas en servicios web JSON	639
Resolución de problemas de despliegue de JSON	639
Resolución de problemas con el asistente de JSON	640
Resolución de problemas relativos a las solicitudes de JSON	641
Respuestas de error devueltas al cliente	642
Códigos de retorno del asistente de JSON	643

Apéndice. Ejemplos de servicios web 645

Aplicación de ejemplo de gestor de catálogos CICS	645
La aplicación base	646
Instalación y configuración de la aplicación base	648
Ejecución de la aplicación de ejemplo con la interfaz BMS	654
Soporte de servicios web para la aplicación de ejemplo	656
Configuración del cliente web	669
Ejecución de la aplicación habilitada para el servicio web.	670
Despliegue de la aplicación de ejemplo.	675
Componentes de la aplicación base	680
Estructuras de archivos y definiciones	688
Ejemplos de JSON.	690
Ejemplo de solicitud de HTTP GET mediante una cadena de consulta	691
Solicitud de ejemplo HTTP con un cuerpo JSON	691

Notices 693

Índice. 699

Acerca de este PDF

Este PDF describe cómo utilizar servicios web en CICS. Está orientado a programadores del sistema responsables de la configuración de CICS para dar soporte a servicios web, y a desarrolladores de aplicaciones responsables de aplicaciones que se desplegarán en un entorno de servicios web. Antes de CICS TS V5.4, este PDF se llamaba "Guía de servicios web".

Para obtener detalles de los términos y notación utilizados en este libro, consulte *Conventions and terminology used in the CICS documentation* en IBM Knowledge Center.

Fecha de este PDF

Este PDF se creó en June 6th 2017.

Capítulo 1. Servicios web y CICS

CICS proporciona soporte para los servicios web.

¿Qué es un servicio web?

Un servicio web tiene una interfaz, que oculta detalles de implementación para que se pueda utilizar independientemente de la plataforma de hardware o software en la que está implementado, e independientemente del lenguaje de programación en el que está escrito. Esta independencia fomenta que las aplicaciones basadas en servicios web sean implementaciones de tecnologías cruzadas, orientadas al componente y no suelen tener conexión directa. Los servicios web se pueden utilizar solos o con otros servicios web para llevar a cabo una agregación compleja o una transacción empresarial.

Servicios web soportados por CICS

CICS soporta dos protocolos de servicio web distintos, los protocolos SOAP y JavaScript Object Notation (JSON). Estos dos protocolos tienen características y ventajas distintas.

Terminología de servicios web

Lenguaje de marcado extensible (Extensible Markup Language - XML)

Estándar para la marcación de documentos, que utiliza una sintaxis genérica para marcar datos con pestañas simples y legibles. El estándar se valida por el World Wide Web Consortium (W3C).

Emisor SOAP inicial

El emisor SOAP que origina un mensaje SOAP en el punto inicial de una vía de acceso de mensaje SOAP.

JavaScript Object Notation (JSON)

Un formato de intercambio de datos ligero que se basa en la notación literal de objetos de JavaScript. JSON es independiente del lenguaje de programación, sin embargo utiliza convenciones de lenguajes entre los que se incluyen C, C++, C#, Java™, JavaScript, Perl, Python.

Esquema JSON

Un documento JavaScript Object Notation que describe la estructura y restringe el contenido de los demás documentos JSON.

RESTful

Relativo a aplicaciones y servicios que se ajustan a las restricciones REST (Representational State Transfer)

Proveedor de servicios

La recopilación de software que proporciona un servicio web.

Aplicación de proveedor de servicios

Una aplicación que se utiliza en un proveedor de servicios. Normalmente, una aplicación de proveedor de servicios proporciona el componente de lógica empresarial de un proveedor de servicios.

Solicitante de servicio

La recopilación de software responsable de la solicitud de un servicio web desde un proveedor de servicios.

Aplicación de solicitante de servicios

Una aplicación que se utiliza en un solicitante de servicio. Normalmente, una aplicación de solicitante de servicio proporciona el componente de lógica empresarial de un solicitante de servicio.

Protocolo de acceso a objetos simple (Simple Object Access Protocol)

Véase SOAP.

SOAP Anteriormente un acrónimo para *Protocolo de acceso a objetos simple (Simple Object Access Protocol)*. Un protocolo de poca importancia para el intercambio de información en un entorno distribuido y descentralizado. Es un protocolo basado en XML que consta de tres partes:

- Un sobre que define un marco de trabajo para describir qué está en el mensaje y cómo procesarlo
- Un conjunto de reglas de codificación para expresar instancias de tipos de datos definidos a nivel de aplicación
- Un convenio para la representación de llamadas y respuestas a procedimientos remotos

SOAP puede utilizarse con otros protocolos, tales como HTTP.

La especificación para SOAP 1.1 se publica en Simple Object Access Protocol (SOAP) 1.1.

La especificación para SOAP 1.2 se publica aquí:

SOAP Versión 1.2 Parte 0: Primer

SOAP Versión 1.2 Parte 1: Infraestructura de mensajería

SOAP Versión 1.2 Parte 2: Adjuntos

Intermediario SOAP

Un nodo SOAP que es un destinatario SOAP y un remitente SOAP accesible desde dentro de un mensaje SOAP. Procesa los bloques de cabecera SOAP dirigidos al mismo y reenvía un mensaje SOAP a un destinatario SOAP final.

Vía de acceso de mensaje SOAP

El conjunto de nodos SOAP a través del que pasa un único mensaje SOAP. Estos nodos incluyen el remitente SOAP inicial, cero o más intermediarios y un destinatario SOAP final.

Nodo SOAP

Lógica de proceso que opera en un mensaje SOAP.

Destinatario SOAP

Un nodo SOAP que acepta un mensaje SOAP.

Emisor SOAP

Un nodo SOAP que transmite un mensaje SOAP.

Destinatario SOAP final

El destinatario SOAP que es un destino final de un mensaje SOAP. Es responsable de procesar el contenido de los bloques de cuerpo y cabecera SOAP destinados a éste.

UDDI Véase Universal Description, Discovery and Integration

Universal Description, Discovery and Integration

Universal Description, Discovery and Integration (UDDI) es una especificación para registros de información basados en la web distribuidos de servicios web. UDDI es también un conjunto de implementaciones de la especificación públicamente accesible que permite a las empresas registrar

información sobre los servicios web que ofrecen para que otras empresas puedan encontrarlos. La especificación se publica en OASIS.

Servicio web

Sistema de software designado para dar soporte a la interacción interoperable de máquina a máquina a través de una red. Tiene una interfaz descrita en formato procesable por la máquina (específicamente, Web Service Description Language o WSDL).

Web Services Addressing

Web Services Addressing (WS-Addressing) proporciona un mecanismo neutral de transporte para direccionar mensajes y servicios web.

Las especificaciones para WS-Addressing se publican aquí:

- Web Services Addressing 1.0 - Clase Core
- Web Services Addressing 1.0 - Enlace SOAP
- Web Services Addressing 1.0 - Metadatos
- Web Services Addressing - Envío

Transacción atómica de servicios web

Una especificación que proporciona la definición de un tipo de coordinación de transacción atómica utilizando para coordinar actividades que tienen una propiedad "todo o nada".

La especificación la publica OASIS en Transacción Atómica de Servicios Web.

Archivo de enlace de servicio web

Un archivo, asociado a un recurso WEBSERVICE, que contiene información que utiliza CICS para correlacionar datos entre mensajes de entrada y salida y estructuras de datos de aplicación.

Descripción de servicio web

Un documento XML por el que un proveedor de servicios comunica las especificaciones para invocar un servicio web para un solicitante de servicio. Las descripciones de servicio web se escriben en WSDL (Web Service Description Language).

Web Service Description Language

Una aplicación XML para describir los servicios web. Está diseñado para separar las descripciones de funciones abstractas ofrecidas por un servicio y los detalles concretos de un servicio, como dónde y cómo se ofrece esa función.

La especificación se publica en Web Services Description Language (WSDL).

Seguridad de servicios web

Un conjunto de mejoras para la mensajería de SOAP que proporciona confidencialidad e integridad de mensaje. La especificación la publica OASIS en Web Services Security: SOAP Message Security 1.0 (WS-Security 2004).

WS-Atomic Transaction

Véase Transacción Atómica de Servicios Web.

Perfil básico WS-I

Un conjunto de especificaciones de servicios web no propietarios, con aclaraciones y modificaciones a esas especificaciones, que, juntas, promueven la interoperabilidad entre diferentes implementaciones de servicios web. La organización de interoperatividad de servicios web

(WS-I) define el perfil y la versión 1.0 está disponible en Organización de interoperatividad de servicios web (WS-I) - Perfil básico 1.0.

WSDL

Véase Web Service Description Language.

WSS Véase Seguridad de servicios web.

XML Extensible Markup Language.

Las especificaciones para XML se publican aquí:

SOAP Versión 1.2 Parte 0: Primer

SOAP Versión 1.2 Parte 1: Infraestructura de mensajería

SOAP Versión 1.2 Parte 2: Adjuntos

Espacio de nombres XML

Una recolección de nombres, identificados por una referencia URI, que se utiliza en documentos XML como tipos de elemento y nombres de atributo.

Esquema XML

Un documento XML que describe la estructura y restringe el contenido de otros documentos XML.

Lenguaje de definición de esquema XML

Una sintaxis XML para escribir esquemas XML, recomendada por World Wide Web Consortium (W3C).

CICS y servicios web SOAP

CICS soporta dos enfoques distintos para el despliegue de las aplicaciones CICS en un entorno de servicios web. Un enfoque permite un despliegue rápido, con la menor cantidad de esfuerzo de programación; el otro enfoque le proporciona una completa flexibilidad y control sobre sus aplicaciones de servicio web, utilizando el código que escribe para ajustarse a sus necesidades concretas. Ambos enfoques están respaldados por una infraestructura que consta de uno o más programas de *interconexiones* y *manejador de mensajes* que operan en solicitudes y respuestas de servicio web.

Cuando despliega aplicaciones CICS en un entorno de servicios web, puede elegir de entre las siguientes opciones:

- Utilice el asistente de servicios web de CICS para que le ayude a desplegar una aplicación con la menor cantidad de esfuerzo de programación.

Por ejemplo, si desea exponer una aplicación existente como servicio web, puede comenzar con una estructura de datos de lenguaje de alto nivel y generar la descripción de servicios web. De lo contrario, si desea comunicarse con un servicio web existente, puede comenzar con su descripción de servicio web y generar una estructura de lenguaje de alto nivel que puede utilizar en el programa.

El asistente de servicios web de CICS también genera los recursos CICS que necesita para desplegar la aplicación. Y cuando se ejecuta la aplicación, CICS transforma la aplicación en un mensaje SOAP en la salida y lo transforma de nuevo en datos de aplicación en la entrada.

- Asuma el control completo sobre el proceso de datos escribiendo su propio código para correlacionar entre los datos de aplicación y el mensaje que fluye entre el proveedor y el solicitante de servicio.

Por ejemplo, si desea utilizar mensajes que no son SOAP dentro de la infraestructura de servicio web, puede escribir su propio código para transformarlo entre el formato de mensaje y el formato utilizado por la aplicación.

Cualquiera que sea el enfoque que sigue, puede utilizar sus propios manejadores de mensajes para llevar a cabo un proceso adicional en los mensajes de repuesta y solicitud o utilizar los manejadores de mensajes proporcionados por CICS diseñados especialmente para ayudarle a procesar mensajes SOAP.

Nodos SOAP

Un nodo SOAP es la lógica de proceso que opera en un mensaje SOAP.

Un nodo SOAP puede realizar estas operaciones:

- Transmitir un mensaje SOAP
- Recibir un mensaje SOAP
- Procesar un mensaje SOAP
- Retransmitir un mensaje SOAP

Un nodo SOAP puede ser de uno de estos tipos:

Emisor SOAP

Un nodo SOAP que transmite un mensaje SOAP.

Destinatario SOAP

Un nodo SOAP que acepta un mensaje SOAP.

Emisor SOAP inicial

El emisor SOAP que origina un mensaje SOAP en el punto inicial de una vía de acceso de mensaje SOAP.

Intermediario SOAP

Un intermediario SOAP es un destinatario SOAP y un emisor SOAP, accesible desde dentro de un mensaje SOAP. Procesa los bloques de cabecera SOAP dirigidos al mismo y actúa para reenviar un mensaje SOAP a un destinatario SOAP final.

Destinatario SOAP final

El destinatario SOAP que es un destino final de un mensaje SOAP. Procesa el contenido del cuerpo SOAP y de cualquier bloque de cabecera SOAP destinado a éste. En algunos casos, es posible que un mensaje SOAP no alcance un destinatario SOAP final; por ejemplo, debido a un problema en un intermediario SOAP.

Una interconexión de proveedor de servicios

En una interconexión de proveedor de servicios, CICS recibe una solicitud, que se pasa a través de una interconexión a un programa de aplicación de destino. La respuesta de la aplicación se devuelve al solicitante de servicio a través de la misma interconexión.

Cuando CICS se encuentra en el rol de solicitante de servicio, realiza las siguientes operaciones:

1. Recibir la solicitud del solicitante de servicio.
2. Examinar la solicitud y extraer el contenido relevante para el programa de aplicación de destino.
3. Invocar al programa de aplicación, pasando los datos extraídos de la solicitud.

4. Cuando el programa de aplicación devuelve el control, construya una respuesta, utilizando los datos devueltos por el programa de aplicación.
5. Envíe una respuesta al solicitante de servicio.

Figura 1 ilustra una interconexión de tres manejadores de mensajes en un valor de proveedor de servicios:

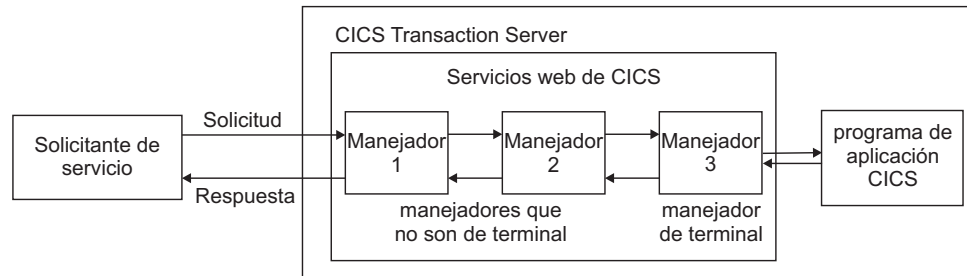


Figura 1. Una interconexión de proveedor de servicios

1. CICS recibe una solicitud del solicitante de servicio. Pasa la solicitud al manejador de mensajes 1.
2. El manejador de mensajes 1 realiza algún proceso y pasa la solicitud al manejador 2 (para ser precisos, devuelve el control a CICS, que maneja la interconexión. CICS pasa el control al siguiente manejador de mensajes).
3. El manejador de mensajes 2 recibe la solicitud del manejador 1, realiza algún proceso y pasa la solicitud al manejador 3.
4. El manejador de mensajes 2 es el controlador de terminal de la interconexión. Utiliza la información en la solicitud para invocar el programa de aplicación. Después, utiliza la salida desde el programa de aplicación para generar una respuesta, que pasa de nuevo al manejador 2.
5. El manejador de mensajes 2 recibe la respuesta del manejador 3, realiza algún proceso y la pasa al manejador 1.
6. El manejador de mensajes 1 recibe la respuesta del manejador 2, realiza algún proceso y devuelve la respuesta al solicitante de servicio.

Una interconexión de solicitante de servicio

En una interconexión de solicitante de servicio, un programa de aplicación crea una solicitud que se pasa a través de la interconexión al proveedor de servicios. La respuesta del proveedor de servicios se devuelve al programa de aplicación a través de la misma interconexión.

Cuando CICS está en el rol del solicitante de servicio, realiza las siguientes operaciones:

1. Utilizar datos proporcionados por el programa de aplicación para construir una solicitud.
2. Enviar la solicitud al proveedor de servicios.
3. Recibir una respuesta del proveedor de servicios.
4. Examinar la respuesta y extraer el contenido relevante para el programa de aplicación original.
5. Devolver el control al programa de aplicación.

Figura 2 ilustra una interconexión de tres manejadores de mensajes en un valor de solicitante de servicio:

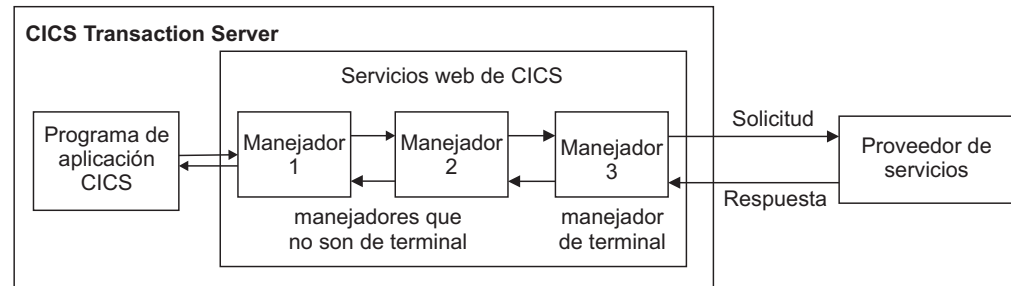


Figura 2. Una interconexión de solicitante de servicio

1. Un programa de aplicación crea una solicitud.
2. El manejador de mensajes 1 recibe la solicitud del programa de aplicación, realiza algún proceso y pasa la solicitud al manejador 2 (para ser precisos, devuelve el control a CICS, que maneja la interconexión. CICS pasa el control al siguiente manejador de mensajes).
3. El manejador de mensajes 2 recibe la solicitud del manejador 1, realiza algún proceso y pasa la solicitud al manejador 3.
4. El manejador de mensajes 3 recibe la solicitud del manejador 2, realiza algún proceso y pasa la solicitud al proveedor de servicios.
5. El manejador de mensajes 3 recibe la respuesta del proveedor de servicios, realiza algún proceso y la pasa al manejador 2.
6. El manejador de mensajes 2 recibe la respuesta del manejador 3, realiza algún proceso y la pasa al manejador 1.
7. El manejador de mensajes 1 recibe la respuesta del manejador 2, realiza algún proceso y la pasa al programa de aplicación.

Interconexiones CICS y SOAP

La interconexión que utiliza CICS para procesar solicitudes y respuestas de servicios web es genérica, en que hay pocas restricciones sobre qué procesos se pueden llevar a cabo en cada manejador de mensajes. Sin embargo, muchas aplicaciones de servicios web, utilizan mensajes SOAP, y cualquier proceso de esos mensajes debe cumplir con la especificación SOAP. Por lo tanto, CICS proporciona programas de *manejador de mensajes SOAP* especiales que pueden ayudarle a configurar la interconexión como un nodo SOAP.

- Una interconexión se puede configurar para su uso en un solicitante de servicio o en un proveedor de servicios:
 - Una interconexión de solicitante de servicio es el emisor SOAP inicial para la solicitud, y el destinatario SOAP final para la respuesta
 - Una interconexión de proveedor de servicios es el destinatario SOAP final para la solicitud, y el emisor SOAP inicial para la respuesta

No puede configurar una interconexión CICS para que funcione como un intermediario SOAP.

- Se puede configurar una interconexión de solicitante de servicio para dar soporte a SOAP 1.1 o SOAP 1.2, pero no ambas. Sin embargo, se puede configurar una interconexión de proveedor de servicios para dar soporte a SOAP 1.1 y SOAP 1.2. En el sistema CICS, puede tener muchas interconexiones, algunas de las cuales soportan SOAP 1.1 o SOAP 1.2 y otras soportan ambas.

- Puede configurar una interconexión CICS para tener más de un manejador de mensajes SOAP.
- Los manejadores de mensajes SOAP proporcionados por CICS puede configurarse para invocar una o más rutinas de manejo de cabeceras escritas por el usuario.
- Los manejadores de mensajes SOAP proporcionados por CICS se pueden configurar para imponer algunos aspectos de conformidad con WS-I Basic Profile Versión 1.1, y para imponer la presencia de cabeceras particulares en el mensaje SOAP.

Los manejadores de mensajes SOAP y sus rutinas de manejo de cabeceras se especifican en el archivo de configuración de la interconexión.

La vía de acceso del mensaje SOAP

La vía de acceso del mensaje SOAP es el conjunto de nodos SOAP a través del cual pasa un único mensaje SOAP, incluido el emisor SOAP inicial, cero o más intermediarios SOAP y un receptor SOAP final

En el caso más sencillo, se transmite un mensaje SOAP entre dos nodos; es decir, desde un *emisor SOAP* a un *receptor SOAP*. Sin embargo, en casos más complejos, los mensajes se pueden procesar por medio de nodos *intermediarios SOAP*, que reciben un mensaje SOAP y, a continuación, lo envían al nodo siguiente. Figura 3 muestra un ejemplo de una vía de acceso de mensaje SOAP, en la que un mensaje SOAP se transmite desde el nodo emisor SOAP inicial al nodo receptor SOAP final, pasando a través de dos nodos intermediarios SOAP en su ruta.

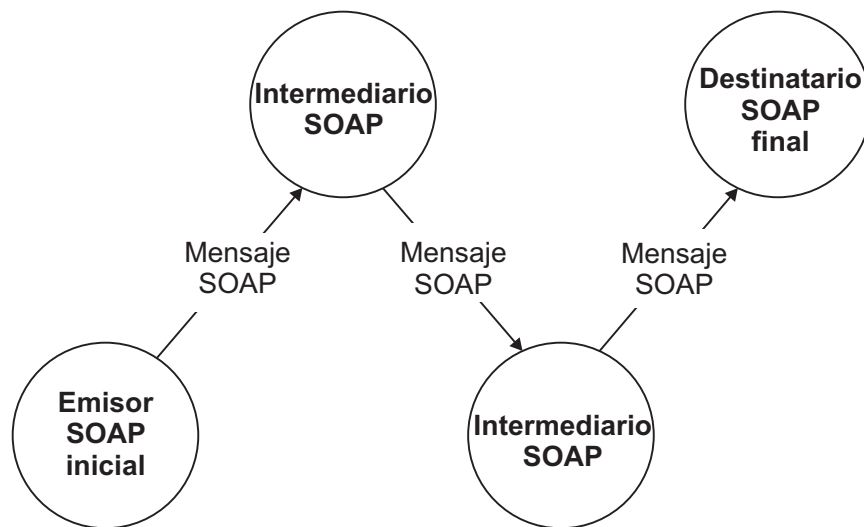


Figura 3. Un ejemplo de vía de acceso de mensaje SOAP

Un intermediario SOAP es un receptor SOAP y un emisor SOAP. Puede, y en algunos casos debe, procesar los bloques de cabecera en el mensaje SOAP, y reenviar el mensaje SOAP hacia su receptor final.

El *receptor SOAP final* es el destino final de un mensaje SOAP. Igual que procesa bloques de cabecera, procesa el cuerpo SOAP. En algunos casos, es posible que un mensaje SOAP no alcance un destinatario SOAP final; por ejemplo, debido a un problema en un intermediario SOAP.

Mensajes SOAP y la estructura de datos de la aplicación

En muchos casos, el asistente de servicios web de CICS puede generar el código para transformar los datos entre una estructura de datos de alto nivel utilizada en un programa de aplicación y el contenido del elemento <Body> de un mensaje SOAP. En estos casos, cuando escribe el programa de aplicación, no necesita analizar o construir el cuerpo SOAP; CICS los hará por usted.

Para transformar los datos, CICS necesita información, en tiempo de ejecución, sobre la estructura de datos de aplicación y sobre el formato de mensajes SOAP. Esta información se contiene en dos archivos:

- Archivo de enlace de servicio web

Este archivo lo genera el asistente de servicios web de CICS desde la estructura de datos de lenguaje de aplicación, utilizando el programa de utilidad DFHLS2WS, o desde una descripción de servicio web, utilizando el programa de utilidad DFHWS2LS. CICS utiliza el archivo de enlace para generar los recursos utilizado por la aplicación de servicio web, y para llevar a cabo la correlación entre la estructura de datos de la aplicación y los mensajes SOAP.

- Descripción de servicio web

Puede ser una descripción de servicio web existente o puede generarse desde una estructura de datos de lenguaje de aplicación, utilizando el programa de utilidad DFHLS2WS. CICS utiliza la descripción de servicio web para realizar una validación completa de mensajes SOAP.

Figura 4 muestra dónde se utilizan estos archivos en un proveedor de servicios.

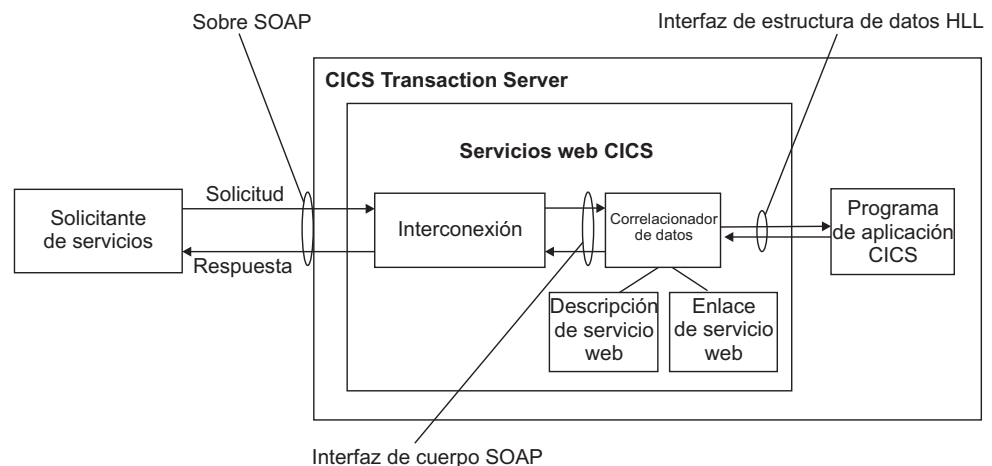


Figura 4. Correlación del cuerpo SOAP con la estructura de datos de aplicación en un proveedor de servicios

Un manejador de mensajes en la interconexión (normalmente, un manejador de mensajes SOAP proporcionado por CICS) elimina el sobre SOAP de una solicitud de entrada y pasa el cuerpo SOAP a la función de correlacionador de datos. Éste utiliza el archivo de enlace de servicio web para correlacionar el contenido del

cuerpo SOAP con la estructura de datos de la aplicación. Si la validación completa del mensaje SOAP está activa, el cuerpo SOAP se valida en la descripción de servicio web. Si hay una respuesta de salida, el proceso se invertirá.

Figura 5 muestra dónde se utilizan estos archivos en un solicitante de servicio.

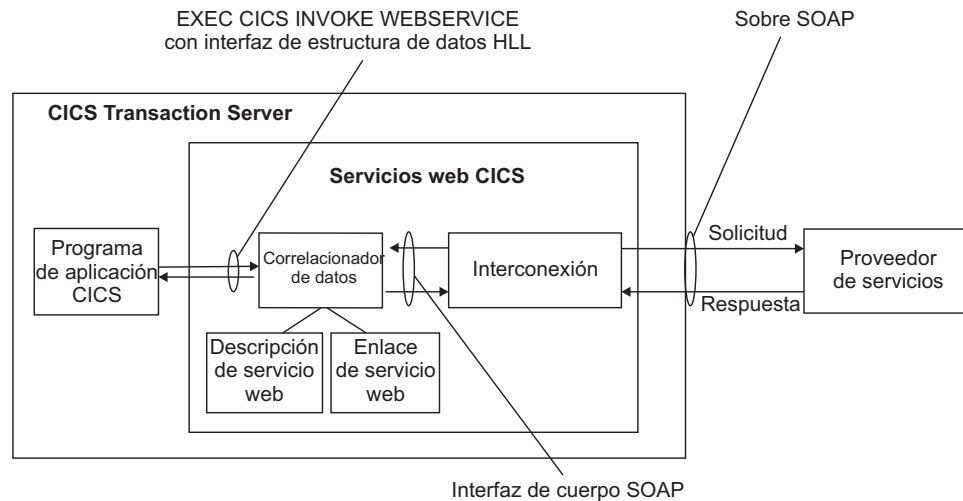


Figura 5. Correlación del cuerpo SOAP con la estructura de datos de aplicación en un solicitante de servicio

Para una solicitud de salida, la función de correlacionador de datos construye un cuerpo SOAP desde la estructura de datos de la aplicación, utilizando información del archivo de enlace de servicio web. Un manejador de mensajes en la interconexión (normalmente, un manejador de mensajes SOAP proporcionado por CICS) añade el sobre SOAP. Si hay una respuesta de entrada, el proceso se invertirá. Si la validación completa del mensaje SOAP está activa, el cuerpo SOAP de entrada se valida en la descripción de servicio web.

En ambos casos, el entorno de ejecución que permite a un programa de aplicación CICS particular operar en un valor de servicios web se define por tres objetos. Son la interconexión, el archivo de enlace de servicio web y la descripción de servicio web. Los tres objetos se definen para CICS como atributos de la definición de recursos WEBSERVICE.

Hay algunas situaciones en las que, aunque está utilizando mensajes SOAP, no puede utilizar la transformación que genera el asistente de servicios web CICS:

- Cuando los mismos datos no se pueden representar en el mensaje SOAP y en el lenguaje de alto nivel.

Todos los lenguajes de alto nivel que soporta CICS, y el esquema XML, soportan varios tipos de datos diferentes. Sin embargo, no hay una correspondencia uno a uno entre los tipos de datos utilizados en lenguajes de alto nivel y los utilizados en el esquema XML, y hay casos en los que los datos se pueden representar en uno pero no en el otro. En estas situaciones, debe tener en cuenta una de las siguientes acciones:

- Cambiar la estructura de datos de aplicación. Esto puede no ser factible ya que puede implicar cambios en el programa de aplicación.

- Construir un programa derivador, que transforma los datos de aplicación en un formato que CICS puede transformar después en un cuerpo de mensaje SOAP. Si hace esto, puede dejar el programa de aplicación sin modificar. En este caso, el soporte de servicio web de CICS interactúa directamente con el programa derivador, y sólo indirectamente con el programa de aplicación.
- Cuando el programa de aplicación está en un lenguaje que no soporta el asistente de servicios web de CICS.
En esta situación, debe tener en cuenta una de las siguientes acciones:
 - Construir un programa derivador que se escribe en uno de los lenguajes que el asistente de servicios web de CICS no soporta (COBOL, PL/I, C o C++).
 - En lugar de utilizar el asistente de servicios web de CICS, escriba su propio programa para realizar la correlación entre los mensajes SOAP y la estructura de datos del programa de aplicación.

WSDL y la estructura de datos de la aplicación

Una descripción de servicio web contiene representaciones abstractas de los mensajes de entrada y salida utilizados por el servicio. CICS utiliza la descripción de servicio web para construir estructuras de datos utilizadas por programas de aplicación. En tiempo de ejecución, CICS realiza la correlación entre las estructuras de datos de la aplicación y los mensajes.

La descripción de un servicio web contiene, entre otras cosas:

- Una o más operaciones
- Para cada operación, un mensaje de entrada y un mensaje de salida opcional
- Para cada mensaje, la estructura del mensaje, definida en términos de tipos de datos XML. Los tipos de datos complejos utilizados en los mensajes se definen en un esquema XML contenido en el elemento <types> dentro de la descripción de servicio web. Los mensajes simples pueden describirse utilizando el elemento <tipos>.

WSDL contiene una definición abstracta de una operación, y los mensajes asociados; no se puede utilizar directamente en un programa de aplicación. Para implementar la operación, un proveedor de servicios debe hacer lo siguiente:

- Debe analizar el WSDL, para entender la estructura de los mensajes
- Debe analizar cada mensaje de entrada y construir el mensaje de salida
- Debe realizar las correlaciones entre el contenido de los mensajes de entrada y de salida, y las estructuras de datos utilizadas en el programa de aplicación

Un solicitante de servicio debe hacer lo mismo para invocar la operación.

Cuando utiliza el asistente de servicios web de CICS, mucho de esto ya está hecho, y puede escribir el programa de aplicación sin una comprensión detallada de WSDL, o de la forma en la que se construyen los mensajes de entrada y de salida.

El asistente de servicios web de CICS consta de dos programas de utilidad:

DFHWS2LS

Este programa de utilidad toma una descripción de servicio web como punto de partida. Utiliza las descripciones de los mensajes y los tipos de datos utilizados en esos mensajes, para construir estructuras de datos de lenguaje de alto nivel que puede utilizar en sus programas de aplicación.

DFHLS2WS

Este programa de utilidad toma la estructura de datos de lenguaje de alto

nivel como punto de partida. Utiliza la estructura para construir una descripción de servicios web que contiene descripciones de mensajes, y los tipos de datos utilizados en esos mensajes derivados de la estructura de lenguaje.

Ambos programas de utilidad generan un archivo de enlace de servicios web que CICS utiliza en el tiempo de ejecución para realizar la correlación entre las estructuras de datos del programa de aplicación y los mensajes SOAP.

Un ejemplo de correlación COBOL a WSDL

Este ejemplo muestra cómo la estructura de datos utilizada en un programa COBOL se representa en la descripción de servicios web que genera el asistente de servicios web de CICS.

Figura 6 muestra una estructura de datos COBOL sencilla:

```
*   Catalogue COMMAREA structure
      03 CA-REQUEST-ID          PIC X(6).
      03 CA-RETURN-CODE         PIC 9(2).
      03 CA-RESPONSE-MESSAGE    PIC X(79).
*   Fields used in Place Order
      03 CA-ORDER-REQUEST.
          05 CA-USERID           PIC X(8).
          05 CA-CHARGE-DEPT      PIC X(8).
          05 CA-ITEM-REF-NUMBER  PIC 9(4).
          05 CA-QUANTITY-REQ     PIC 9(3).
          05 FILLER              PIC X(888).
```

Figura 6. Definición de registro COBOL de un mensaje de entrada definido en WSDL

Los elementos clave del correspondiente fragmento de la descripción de servicios web se muestra en Figura 7 en la página 13:

```

<xsd:sequence>
  <xsd:element name="CA-REQUEST-ID" nillable="false">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:length value="6"/>
        <xsd:whiteSpace value="preserve"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:element>
  <xsd:element name="CA-RETURN-CODE" nillable="false">
    <xsd:simpleType>
      <xsd:restriction base="xsd:short">
        <xsd:maxInclusive value="99"/>
        <xsd:minInclusive value="0"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:element>
  <xsd:element name="CA-RESPONSE-MESSAGE" nillable="false">
    ...
  </xsd:element>
  <xsd:element name="CA-ORDER-REQUEST" nillable="false">
    <xsd:complexType mixed="false">
      <xsd:sequence>
        <xsd:element name="CA-USERID" nillable="false">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:length value="8"/>
              <xsd:whiteSpace value="preserve"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element name="CA-CHARGE-DEPT" nillable="false">
          ...
        </xsd:element>
        <xsd:element name="CA-ITEM-REF-NUMBER" nillable="false">
          ...
        </xsd:element>
        <xsd:element name="CA-QUANTITY-REQ" nillable="false">
          ...
        </xsd:element>
        <xsd:element name="FILLER" nillable="false">
          ...
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:sequence>

```

Figura 7. Fragmento de WSDL derivado de una estructura de datos COBOL

WSDL y patrones de intercambio de mensajes

Un documento WSDL 2.0 contiene un patrón de intercambio de mensajes (MEP) que define la forma en la que los mensajes SOAP 1.2 se intercambian entre el solicitante de servicio web y el proveedor de servicios web.

CICS soporta cuatro de ocho patrones de intercambio de mensajes definidos en la especificación *WSDL 2.0 Part 2: Adjuncts* y la especificación *WSDL 2.0 Part 2: Additional MEPs* para las aplicaciones de proveedor de servicios y de solicitante de servicio. Se soportan los siguientes MEPs:

Sólo entrada

Se envía un mensaje de solicitud al proveedor de servicio web, pero el proveedor no tiene permiso para enviar ningún tipo de respuesta al solicitante de servicio web.

- En la modalidad de proveedor, cuando CICS recibe un mensaje de solicitud de un servicio web que utiliza el MEP Sólo entrada, no devuelve un mensaje de respuesta. El contenedor DFHNORESPONSE se

coloca en el canal del manejador SOAP para indicar que la interconexión no debe enviar un mensaje de respuesta.

- En la modalidad de solicitante, CICS envía el mensaje de solicitud al proveedor de servicios y no espera una respuesta.

Entrada-salida

Se envía un mensaje de solicitud al proveedor de servicios web y se devuelve un mensaje de respuesta al solicitante de servicio web. El mensaje de respuesta puede ser un mensaje SOAP normal o un error SOAP.

- En la modalidad de proveedor, cuando CICS recibe un mensaje de solicitud de un servicio web que utiliza el MEP Entrada-salida, devuelve un mensaje de respuesta al solicitante.
- En la modalidad de solicitante, CICS envía el mensaje de solicitud y espera una respuesta. Esta respuesta es un mensaje de respuesta normal o un mensaje de error SOAP. El periodo de tiempo que CICS espera una respuesta se configura en la interconexión y se aplica a todos los servicios web utilizando esa interconexión. Si la solicitud supera el tiempo de espera antes de que CICS reciba una respuesta, se devuelve un error a la aplicación de solicitante de servicio.

Entrada, opcional salida

Se envía un mensaje de solicitud al proveedor de servicios web y, opcionalmente, se devuelve un mensaje de respuesta al solicitante de servicios web. Si hay una respuesta, puede ser un mensaje SOAP normal o un error SOAP.

- En modalidad de proveedor, la decisión sobre si devolver un mensaje de respuesta SOAP, un error SOAP, o no devolver respuesta, tiene lugar en tiempo de ejecución y depende de la lógica de la aplicación de proveedor de servicios. Si CICS no envía una respuesta al solicitante de servicio web, el contenedor DFHNORESPONSE se coloca en el canal de manejador SOAP para indicar que la interconexión no debe enviar un mensaje de respuesta. Si no se envía un mensaje, la aplicación de proveedor de servicios debe suprimir el contenedor DFHWS-DATA del canal.
- En modalidad de solicitante, CICS envía un mensaje de solicitud y espera una respuesta del solicitante de servicio web. Si la solicitud supera el tiempo de espera antes de que se reciba una respuesta, CICS supone que el mensaje se ha recibido correctamente y que el proveedor no necesita enviar una respuesta. El periodo de tiempo que CICS espera una respuesta se configura en la interconexión y se aplica a todos los servicios web utilizando esa interconexión.

Sólido sólo entrada

Se envía un mensaje de solicitud al proveedor de servicios web, y sólo se devuelve un mensaje de respuesta al solicitante de servicio web si se produce un error. Si se produce un error, se envía un mensaje de error SOAP al solicitante.

- En modalidad de proveedor, si la interconexión pasa correctamente el mensaje de solicitud a la aplicación, se coloca un contenedor DFHNORESPONSE en el canal de manejador SOAP para indicar que la interconexión no debe enviar un mensaje de respuesta. Si se produce un error en la interconexión, se devuelve un mensaje de error SOAP al solicitante.
- En la modalidad de solicitante, CICS envía el mensaje de solicitud al proveedor de servicios web y espera un periodo de tiempo especificado

antes de que se exceda el tiempo de espera. El periodo de tiempo que CICS espera una respuesta se configura en la interconexión y se aplica a todos los servicios web utilizando esa interconexión. Si hay un tiempo de espera, CICS supone que el mensaje de solicitud se ha recibido correctamente.

Para obtener más información sobre los patrones de intercambio de mensajes en WSDL 2.0, consulte las siguientes especificaciones W3C:

- *WSDL 2.0 Part 2: Adjuncts*: .
- *WSDL 2.0 Part 2: Additional MEPs*: .

Archivo de enlace de servicio web

El *archivo de enlace de servicio web* contiene información que utiliza CICS para correlacionar datos entre mensajes de entrada y de salida, y estructuras de datos de aplicación.

Una descripción de servicio web contiene representaciones abstractas de mensajes de entrada y salida utilizados por el servicio. Cuando se ejecuta una aplicación de solicitante de servicio o de proveedor de servicios, CICS necesita información sobre cómo el contenido de los mensajes se correlaciona con las estructuras de datos utilizadas por la aplicación. Esta información se contiene en un archivo de enlace de servicio web.

Los archivos de enlace de servicio web se crean:

- Mediante el programa de utilidad DFHWS2LS cuando se generan estructuras de lenguaje desde WSDL.
- Mediante el programa de utilidad DFHLS2WS cuando se genera WSDL desde una estructura de lenguaje.

Durante el tiempo de ejecución, CICS utiliza información en el archivo de enlace de servicio web para realizar la correlación entre las estructuras de datos de aplicación y los mensajes SOAP. Los archivos de enlace de servicio web se definen para CICS en el atributo WSBIND del recurso WEBSERVICE.

Estándares externos

El soporte CICS para servicios web se adecua a varios estándares y especificaciones del sector.

Arquitectura y formato de mensaje SOAP

SOAP es un protocolo para el intercambio de información en un entorno distribuido. Los mensajes SOAP están codificados como documentos XML y pueden intercambiarse utilizando varios protocolos subyacentes.

Anteriormente, un acrónimo para *Protocolo de acceso a objetos simple*, SOAP fue desarrollado por World Wide Web Consortium (W3C) y está definido en los siguientes documentos emitidos por W3C. Consulte estos documentos para obtener información completa y autorizada sobre SOAP.

Simple Object Access Protocol (SOAP) 1.1 (Nota W3C)

SOAP Versión 1.2 Parte 0: Primer (recomendación W3C)

SOAP Versión 1.2 Parte 1: Infraestructura de mensajería (recomendación W3C)

SOAP Versión 1.2 Parte 2: Adjuntos (recomendación W3C)

Las especificaciones SOAP describen un modelo de procesamiento distribuido en el que un *mensaje SOAP* se pasa entre *nodos SOAP*. El mensaje se origina en un *emisor SOAP* y se envía a un *destinatario SOAP*. Entre el emisor y el destinatario, el mensaje se puede procesar por uno o varios *intermediarios SOAP*.

Un mensaje SOAP es una transmisión unidireccional entre nodos SOAP, desde un emisor SOAP a un destinatario SOAP, pero los mensajes se pueden combinar para construir interacciones más complejas, como solicitud y respuesta y conversaciones de igual a igual.

Las especificaciones también incluyen esta información:

- Un conjunto de normas para expresar instancias de tipos de datos definidos por la aplicación.
- Un convenio para representar las llamadas a procedimiento remoto (RPC) y sus respuestas.

Arquitectura de servicios web SOAP

La arquitectura de servicios web SOAP se basa en interacciones entre tres componentes: un proveedor de servicios, un solicitante de servicio y un registro de servicio opcional.

El proveedor de servicios

La recopilación de software que proporciona un servicio web.

- El programa de aplicación
- El middleware
- La plataforma en la que se ejecutan

El solicitante de servicio

La recopilación de software responsable de la solicitud de un servicio web desde un proveedor de servicios.

- El programa de aplicación
- El middleware
- La plataforma en la que se ejecutan

El registro de servicio

El registro de servicio es una ubicación local donde los proveedores de servicios pueden publicar sus descripciones de servicios y donde los solicitantes de servicio pueden encontrar esas descripciones de servicio.

El registro es un componente opcional de la arquitectura de servicios web porque los proveedores y solicitantes de servicio pueden comunicarse sin él en muchas situaciones. Por ejemplo, la organización que proporciona un servicio puede distribuir la descripción de servicio directamente a los usuarios del servicio de varias maneras, incluso ofreciendo el servicio como una descarga desde un sitio FTP.

La utilización de un registro de servicio ofrece varias ventajas al solicitante y al proveedor; por ejemplo, utilizar el IBM® WebSphere Service Registry and Repository (WSRR) puede ayudar al solicitante a encontrar servicios más rápidamente y puede ayudar al proveedor a imponer la versión de control de los servicios que se están ofreciendo.

CICS proporciona soporte directo para la implementación de los componentes de proveedor de servicios y solicitante de servicio. Sin embargo, necesita software adicional para desplegar un registro de servicio en CICS. Si utiliza IBM WebSphere

Service Registry and Repository (WSRR), CICS proporciona soporte para WSRR a través del asistente de servicios web. Sino, puede desplegar un registro de servicio en otra plataforma.

Interacciones entre un proveedor de servicios, un solicitante de servicio y un registro de servicio

Las interacciones entre el proveedor de servicios, el solicitante de servicio y el registro de servicio implica las siguientes operaciones:

Publicar

Cuando se utiliza un registro de servicios, un proveedor de servicios publica su descripción de servicio en un registro de servicio para que lo busque el solicitante de servicio.

Buscar

Cuando se utiliza un registro de servicios, un solicitante de servicio busca la descripción del servicio en el registro.

Enlazar

El solicitante de servicio utiliza la descripción de servicio para enlazar con el proveedor de servicios e interactuar con la implementación de servicio web.



Figura 8. Componentes e interacciones de servicios web

Descripción de servicio web:

Una descripción de servicio web es un documento por el que el *proveedor de servicios* comunica las especificaciones para iniciar el servicio web para el *solicitante de servicio*. Las descripciones de servicios web se expresan en la aplicación XML denominada WSDL (Web Service Description Language).

La descripción de servicio describe el servicio web de tal manera para minimizar la cantidad de conocimiento compartido y de programación personalizada que se necesita para garantizar la comunicación entre el proveedor de servicios y el

solicitante de servicio. Por ejemplo, ni el solicitante ni el proveedor tendrán que saber la plataforma en el que el otro se ejecuta ni el lenguaje de programación en el que el otro está escrito.

Una descripción de servicio puede ajustarse a la especificación WSDL 1.1 o WSDL 2.0. Cada una tiene diferencias en terminología y elementos principales que se pueden incluir en la descripción de servicio. En la siguiente información se utilizan elementos y terminología de WSDL 1.1 para explicar la finalidad de la descripción de servicio.

La estructura de WSDL permite a la descripción de servicio particionarse en dos definiciones:

- Una *definición de interfaz de servicio* abstracta que describe las interfaces de servicio y hace posible que escriba programas que implementan e inician el servicio.
- Una *definición de implementación de servicio* concreta que describe la ubicación en la red (o *punto final*) del servicio web de un proveedor y otros detalles específicos de la implementación. Permite a un solicitante de servicio conectarse al proveedor de servicios.

Consulte el apartado Figura 9 en la página 19.

Un documento WSDL 1.1 utiliza los siguientes elementos principales en la definición de servicios de red:

<types>

Contenedor para definiciones de tipos de datos que utiliza algún sistema de tipo (como el esquema XML). Define los tipos de datos utilizados dentro del mensaje. El elemento <types> no es necesario cuando todos los mensajes constan de tipos de datos simples.

<message>

Especifica qué tipos de datos XML se utilizan para definir los parámetros de entrada y salida de una operación.

<portType>

Define el conjunto de operaciones soportadas por uno o varios puntos finales. Dentro de un elemento <portType>, cada operación se describe por un elemento <operation>.

<operation>

Especifica qué mensajes XML pueden aparecer en los flujos de datos de entrada y salida. Una operación es comparable con una firma de método en un lenguaje de programación.

<binding>

Describe el protocolo, formato de datos, seguridad y otros atributos para un elemento <portType> particular.

<port>

Especifica la dirección de red de un punto final y lo asocia a un elemento <binding>.

<service>

Define el servicio web como una recolección de puntos finales relacionados. Un elemento <service> contiene uno o varios elementos <port>.

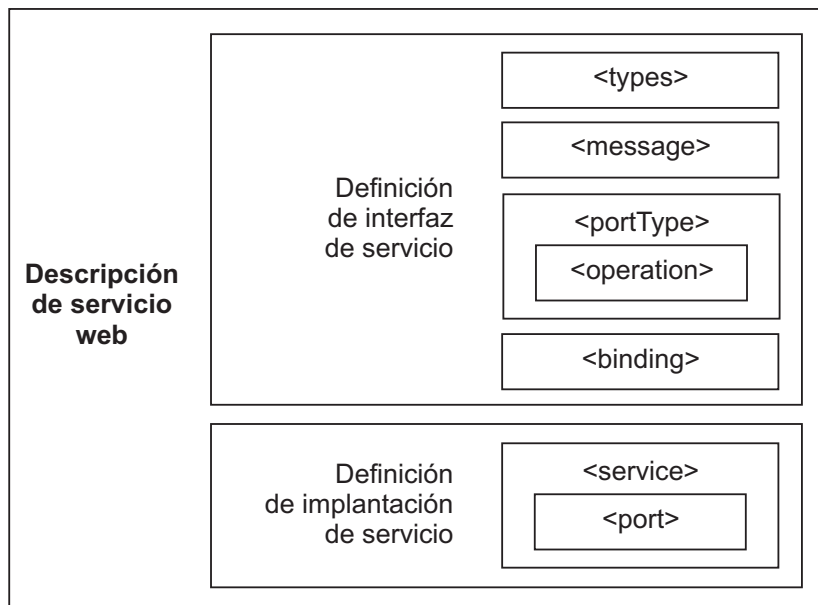


Figura 9. Estructura de una descripción de servicio web

Como puede particionar la descripción de servicio web, puede dividir la responsabilidad para crear una descripción de servicio completa. Igual que una ilustración, considere un servicio como definido por un cuerpo de estándares para su uso en un sector e implementado por compañías individuales de ese sector:

- El cuerpo de estándares proporcionar una definición de interfaz de servicio, que contiene los siguientes elementos:
 - <types>
 - <message>
 - <portType>
 - <binding>
- Un proveedor de servicios que desea ofrecer una implementación del servicio proporciona una definición de implementación de servicio que contiene los siguientes elementos:
 - <port>
 - <service>

Publicación de servicios:

Puede publicar una descripción de servicio utilizando varios mecanismos diferentes. Cada mecanismo es adecuado para utilizarlo en diferentes situaciones. CICS soporta el uso de IBM WebSphere Service Registry and Repository (WSRR) para publicar descripciones de servicio. Sino, puede utilizar otros métodos para publicar una descripción de servicio.

WSSR CICS soporta el uso de WSRR para publicar descripciones de servicio. Para obtener más información sobre el soporte que proporciona CICS para WSSR, "Interoperabilidad entre el asistente de servicios web y WSRR" en Information Center.

Cualquiera de los siguientes mecanismos, ninguno de los cuales está soportado directamente por CICS, se puede utilizar con CICS para publicar descripciones de servicio:

Publicación directa

Este mecanismo es el más sencillo para publicar descripciones de servicio; el proveedor de servicios envía la descripción de servicio directamente al solicitante de servicio, utilizando un archivo adjunto de correo electrónico, un sitio FTP o una distribución en CD ROM.

DISCO

Estos protocolos de propiedad proporcionan un mecanismo de publicación dinámico. El solicitante de servicio utiliza un sencillo mecanismo HTTP GET para recuperar una descripción de servicio web desde una ubicación de red especificada por el proveedor de servicios e identificada con una URL.

Universal Description, Discovery and Integration (UDDI)

Una especificación para registros de información basados en la web distribuidos de servicios web. UDDI es también un conjunto de implementaciones de la especificación públicamente accesible que permite a las empresas registrar información sobre los servicios web que ofrecen para que otras empresas puedan encontrarlos.

Se puede publicar una descripción de servicio en más de un formato si es necesario.

Estructura de un mensaje SOAP

Un mensaje SOAP se codifica como un documento XML, que consta de un elemento <Envelope>, que contiene un elemento <Header> opcional y un elemento <Body> obligatorio. El elemento <Fault>, contenido en <Body>, se utiliza para informar de errores.

El sobre SOAP

El <Envelope> SOAP es el elemento raíz de todos los mensajes SOAP. Contiene dos elementos hijo, una <Header> opcional y un <Body> obligatorio.

La cabecera SOAP

La <Header> SOAP es un subelemento opcional del sobre SOAP. Se utiliza para pasar información relacionada con la aplicación que van a procesar los nodos SOAP junto con la vía de acceso a mensajes.

Cuerpo SOAP

El <Body> SOAP es un subelemento obligatorio del sobre SOAP. Contiene información pensada para el destinatario final del mensaje.

Error SOAP

El <Fault> SOAP es un subelemento del cuerpo SOAP, que se utiliza para la notificación de errores.

Con la excepción del elemento <Fault>, contenido en el <Body> de un mensaje SOAP, los elementos XML de <Header> y <Body> se definen mediante las aplicaciones que hacen uso de ellos. Sin embargo, la especificación SOAP impone algunas restricciones en su estructura.

Figura 10 en la página 21 muestra los elementos principales de un mensaje SOAP.

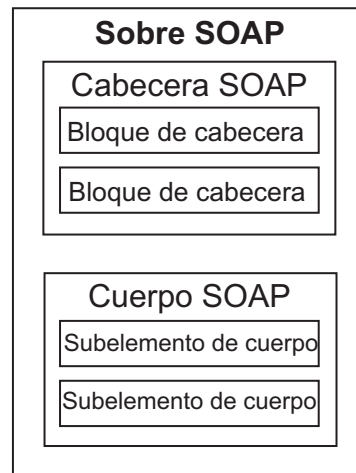


Figura 10. Estructura de un mensaje SOAP

Figura 11 en la página 22 es un ejemplo de un mensaje SOAP que contiene bloques de cabecera (los elementos `<m:reservation>` y `<n:passenger>`) y un cuerpo (que contiene los elementos `<p:itinerary>` y `<q:lodging>`).

```

<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <m:reservation xmlns:m="http://travelcompany.example.org/reservation"
      env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
      env:mustUnderstand="true">
      <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</m:reference>
      <m:dateAndTime>2001-11-29T13:20:00.000-05:00</m:dateAndTime>
    </m:reservation>
    <n:passenger xmlns:n="http://mycompany.example.com/employees"
      env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
      env:mustUnderstand="true">
      <n:name>Åke Jógvan Øyvind</n:name>
    </n:passenger>
  </env:Header>
  <env:Body>
    <p:itinerary
      xmlns:p="http://travelcompany.example.org/reservation/travel">
      <p:departure>
        <p:departing>New York</p:departing>
        <p:arriving>Los Angeles</p:arriving>
        <p:departureDate>2001-12-14</p:departureDate>
        <p:departureTime>late afternoon</p:departureTime>
        <p:seatPreference>aisle</p:seatPreference>
      </p:departure>
      <p:return>
        <p:departing>Los Angeles</p:departing>
        <p:arriving>New York</p:arriving>
        <p:departureDate>2001-12-20</p:departureDate>
        <p:departureTime>mid-morning</p:departureTime>
        <p:seatPreference/>
      </p:return>
    </p:itinerary>
    <q:lodging
      xmlns:q="http://travelcompany.example.org/reservation/hotels">
      <q:preference>none</q:preference>
    </q:lodging>
  </env:Body>
</env:Envelope>

```

Figura 11. Ejemplo de un mensaje SOAP 1.2

La cabecera SOAP:

La <Header> SOAP es un elemento opcional en un mensaje SOAP. Se utiliza para pasar información relacionada con la aplicación que van a procesar los nodos SOAP junto con la vía de acceso a mensajes.

Los elementos hijo inmediatos del elemento <Header> se denominan bloques de cabecera. Un bloque de cabecera es un elemento XML definido por la aplicación. Representa una agrupación lógica de datos que se pueden dirigir a nodos SOAP que pueden encontrarse en la ruta de un mensaje desde el remitente al destinatario final.

Los bloques de cabecera SOAP se pueden procesar por nodos intermediarios SOAP y por el nodo de destinatario final SOAP. Sin embargo, en una aplicación real, no cada nodo procesa cada bloque de cabecera. Más bien, cada nodo se diseña normalmente para procesar bloques de cabecera particulares y, en cambio, cada bloque de cabecera está pensado para que lo procesen nodos particulares.

La cabecera SOAP permite que se añadan características a un mensaje SOAP de forma descentralizada sin un acuerdo anterior entre las partes en comunicación. SOAP define unos cuantos atributos que se pueden utilizar para indicar cuál

tratará con una características y si es opcional u obligatorio. Esta información de "control" incluye, por ejemplo, pasar información contextual o de directivas relacionada con el proceso del mensaje. De esta manera, un mensaje SOAP se puede ampliar de una forma específica para la aplicación.

Aunque los bloques de cabecera están definidos por la aplicación, los atributos definidos por SOAP en los bloques de cabecera indican cómo los nodos SOAP van a procesar los bloques de cabecera. Tenga en cuenta estos atributos importantes:

encodingStyle

Indica las normas utilizadas para codificar las partes de un mensaje SOAP. SOAP define un conjunto de reglas más limitado para codificar datos que la flexible codificación que permite XML.

rol (SOAP 1.2)

actor (SOAP 1.1)

En SOAP 1.2, el atributo **rol** especifica si un nodo particular opera en un mensaje. Si el rol especificado para el nodo coincide con el atributo **role** del bloque de cabecera, el nodo procesa la cabecera. Si los roles no coinciden, el nodo no procesa el bloque de cabecera. En SOAP 1.1, el atributo **actor** tiene la misma función.

Los roles se pueden definir por la aplicación y se designado por un URI. Por ejemplo, <http://example.com/Log> puede designar el rol de un nodo que realiza el registro. Los bloques de cabecera que va a procesar este nodo especifican `env:role="http://example.com/Log"`, donde el prefijo de espacio de nombres `env` se asocia al nombre del espacio de nombres SOAP de <http://www.w3.org/2003/05/soap-envelope>.

La especificación SOAP 1.2 define tres roles estándar además de los que están definidos por la aplicación:

<http://www.w3.org/2003/05/soap-envelope/none>

Ninguno de los nodos SOAP de la ruta del mensaje procesarán el bloque de cabecera directamente. Los bloques de cabecera con este rol se pueden utilizar para transportar los datos necesarios para procesar otros bloques de cabecera SOAP.

<http://www.w3.org/2003/05/soap-envelope/next>

Se espera que todos los nodos SOAP en la ruta del mensaje examinen el bloque de cabecera, siempre que un nodo no haya eliminado la cabecera antes en la ruta del mensaje.

<http://www.w3.org/2003/05/soap-envelope/ultimateReceiver>

Está previsto que sólo el nodo destinatario final examine el bloque de cabecera.

mustUnderstand

Este atributo se utiliza para asegurar que los nodos SOAP no ignoran los bloques de cabecera que son importantes para el propósito general de la aplicación. Si un nodo SOAP determina, utilizando el atributo **rol** o **actor**, que procesará un bloque de cabecera, y el atributo **mustUnderstand** tiene un valor de "true", el nodo debe procesar el bloque de cabecera de forma coherente con su especificación o no procesarlo (y generar un error). Pero si el atributo tiene un valor de "false", el nodo no está obligado a procesar el bloque de cabecera.

En efecto, el atributo **mustUnderstand** indica si el proceso del bloque de cabecera es obligatorio o es opcional.

El atributo **mustUnderstand** tiene estos valores:

true (SOAP 1.2)

1 (SOAP 1.1)

El nodo debe procesar el bloque de cabecera de una forma coherente con su especificación o no procesarlo (y generar un error).

false (SOAP 1.2)

0 (SOAP 1.1)

El nodo no está obligado a procesar el bloque de cabecera.

relé (sólo SOAP 1.2)

Cuando un nodo intermediario SOAP procesa un bloque de cabecera, lo elimina del mensaje SOAP. De forma predeterminada, también elimina los bloques de cabecera que ha ignorado, porque el atributo **mustUnderstand** tenía un valor de "false". Sin embargo, cuando se especifica el atributo **relé** con un valor de "true", el nodo retiene el bloque de cabecera no procesado en el mensaje.

Cuerpo SOAP:

El <Body> es el elemento obligatorio en el sobre SOAP en el que se lleva la información de extremo a extremo principal transmitida en un mensaje SOAP.

El elemento <Body> y sus elementos hijos asociados se utilizan para intercambiar información entre el remitente SOAP inicial y el destinatario SOAP final. SOAP define un elemento hijo para <Body>: el elemento <Fault>, que se utiliza para notificar errores. El servicio web define otros elementos de <Body> que utiliza.

Error SOAP:

El elemento de SOAP <Fault> contiene información de estado y error en el mensaje SOAP.

Si se produce un error en un servicio web, se devuelve un mensaje de error al cliente. La estructura básica del mensaje de error se define en las especificaciones SOAP. Cada mensaje de error puede incluir el XML que describe la condición de error específica. Por ejemplo, si se produce una terminación anómala de la aplicación en un servicio web CICS, se devuelve un mensaje de error al cliente que informa de la terminación anómala.

CICS puede enviar diferentes tipos de mensaje de error:

- Los mensajes de error SOAP estándar se definen mediante especificaciones SOAP o una de las especificaciones de servicio web que se soportan en CICS. Los errores informan de condiciones de error comunes, como sobres SOAP mal formados.
- Los mensajes de error SOAP de aplicación se generan utilizando los mandatos de API **EXEC CICS SOAPFAULT** en respuesta a las condiciones detectadas o gestionadas por la aplicación. La estructura de estos mensajes de error es conocida por la aplicación, pero no por CICS.
- Los mensajes de error del manejador SOAP se generan mediante programas de manejador SOAP en respuesta a un manejo de errores general en CICS. Por ejemplo, los programas de manejador SOAP envían errores SOAP para terminaciones anómalas, errores de análisis XML y otros errores comunes.
- Los mensajes de error del manejador de aplicación se generan mediante los manejadores de aplicación SOAP de CICS en respuesta a la búsqueda de errores

al procesar el cuerpo de un mensaje SOAP. Estos errores se producen durante el proceso de transformación del XML en datos de aplicación binarios o cuando se genera la respuesta.

Si se produce un error, el elemento de SOAP `<Fault>` debe ser una entrada de cuerpo y no debe estar presente más de una vez en un elemento `<Body>`. Los elementos XML dentro del elemento de SOAP `<Fault>` son diferentes en SOAP 1.1 y SOAP 1.2.

SOAP 1.1

En SOAP 1.1, el elemento de SOAP `<Fault>` contiene los siguientes elementos:

`<faultcode>`

El elemento `<faultcode>` es un elemento obligatorio en el elemento `<Fault>`. Proporciona información sobre el error en un formato que el software puede procesar. SOAP define un pequeño conjunto de códigos de error SOAP que cubren errores SOAP básicos, que las aplicaciones puede ampliar.

`<faultstring>`

El elemento `<faultstring>` es un elemento obligatorio dentro del elemento `<Fault>`. Proporciona información sobre el error en un formato que el usuario puede entender.

`<faultactor>`

El elemento `<faultactor>` contiene el URI del nodo SOAP que ha generado el error. Un nodo SOAP que no es el destinatario SOAP final debe incluir el elemento `<faultactor>` cuando crea un error. Un destinatario SOAP final no está obligado a incluir este elemento, pero puede hacerlo.

`<detail>`

El elemento `<detail>` incluye información de error específica de la aplicación relacionada con el elemento `<Body>`. Debe estar presente si el contenido del elemento `<Body>` no se ha procesado satisfactoriamente. No se debe utilizar para llevar información sobre el error perteneciente a las entradas de cabecera. La información de error detallada perteneciente a entradas de cabecera debe transportarse dentro de entradas de cabecera.

SOAP 1.2

En SOAP 1.2, el elemento de SOAP `<Fault>` contiene los siguientes elementos:

`<Code>`

El elemento `<Code>` es un elemento obligatorio dentro del elemento `<Fault>`. Proporciona información sobre el error en un formato que el software puede procesar. Contiene un elemento `<Value>` y un elemento `<Subcode>` opcional.

`<Reason>`

El elemento `<Reason>` es un elemento obligatorio dentro del elemento `<Fault>`. El elemento `<Reason>` contiene uno o más elementos `<Text>`, cada uno de los cuales contiene información sobre el error en un lenguaje nativo diferente.

`<Node>`

El elemento `<Node>` contiene el URI del nodo SOAP que ha generado el error. Un nodo SOAP que no es el destinatario SOAP final debe incluir el

elemento <Node> cuando crea un error. Un destinatario SOAP final no está obligado a incluir este elemento, pero puede hacerlo.

<Role>

El elemento <Role> contiene un URI que identifica el rol que el nodo desempeñaba en el momento en que se produjo el error.

<Detail>

El elemento <Detail> es un elemento opcional, que contiene información de errores específica de la aplicación relacionada con los códigos de error SOAP que describen el error. La presencia del elemento <Detail> no tiene importancia en relación a las partes del mensaje SOAP de error que se han procesado.

Esquemas y ejemplo de error SOAP

El ejemplo siguiente muestra un mensaje de error SOAP generado por el manejador de aplicación DFHPITP, al procesar el cuerpo de un mensaje SOAP.

```
<SOAP-ENV:Fault xmlns="">
  <faultcode>SOAP-ENV:Server</faultcode>
  <faultstring>Conversion to SOAP failed</faultstring>
  <detail>
    <CICSFault xmlns="http://www.ibm.com/software/htp/cics/WSFault">
      DFHPI1008 25/01/2010 14:16:50 IYCWZCFU 00340 XML
      generation failed because of incorrect input
      (CONTAINER_NOT_FOUND container name) for WEBSERVICE
      servicename.
    </CICSFault>
  </detail>
</SOAP-ENV:Fault>
```

La mayoría del contenido del ejemplo es común para todos los mensajes de error. El elemento <Detail> contiene información exclusiva que describe el problema que se ha encontrado el manejador de aplicación. Este mensaje de error específico contiene una copia de un mensaje de error escrito en los registros de mensajes CICS. Si desea analizar los errores SOAP de manejador de aplicación de forma programada, utilice el siguiente esquema XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.ibm.com/software/htp/cics/WSFault"
  xmlns:tns="http://www.ibm.com/software/htp/cics/WSFault"
  elementFormDefault="qualified">
  <element name="CICSFault" type="string">
    <annotation>
      <documentation>
        The value of this element is a text string that describes a
        problem encountered during the processing of the Body of a
        SOAP message.
      </documentation>
    </annotation>
  </element>
</schema>
```

Los mensajes de error de propósito general son más complicados porque la sección <Detail> se puede estructurar de varias formas diferentes. Si desea analizar los errores de manejador SOAP de forma programada, utilice el esquema XML que se proporciona en *usshome/schemas/soapfault/soapfault.xsd*, donde *usshome* es el valor del parámetro de inicialización del sistema **USSHOME**.

Planificación para utilizar servicios web SOAP

Antes de planificar el uso de los servicios web SOAP en CICS, necesita tener en cuenta estas preguntas para cada aplicación.

Antes de empezar

¿Desea desplegar la aplicación CICS en el rol de un proveedor de servicios o un solicitante de servicio?

Puede tener un par de aplicaciones que desea conectar utilizando el soporte CICS para servicios web. En este caso, una aplicación será el proveedor de servicios; la otra será el solicitante de servicio.

¿Tiene previsto utilizar sus programas de aplicación existentes o escribir otros nuevos?

Si sus aplicaciones existentes se han diseñado con una interfaz bien definida en cuanto a la lógica empresarial, probablemente podrá utilizarlas en una configuración de servicios web, como proveedor de servicios o como solicitante de servicio. Sin embargo, en la mayoría de los casos, tendrá que escribir un programa derivador que conecte la lógica empresarial con la lógica de servicios.

Si tiene previsto escribir aplicaciones nuevas, intente mantener la lógica empresarial separada de la lógica de servicios web y, una vez más, tendrá que escribir un programa derivador para proporcionar esta separación. Sin embargo, si la aplicación está diseñada pensando en los servicios web, resultará más sencillo escribir el programa derivador.

¿Desea utilizar mensajes SOAP?

SOAP es fundamental para la arquitectura de servicios web y parte del soporte que proporciona en CICS asume que utilizará SOAP. Sin embargo, puede haber situaciones en las que desee utilizar otros formatos de mensaje. Por ejemplo, puede que haya desarrollado sus propios formatos de mensaje que desea desplegar con la infraestructura de servicios web CICS. Puede hacer esto con CICS, pero no podrá utilizar algunas de las funciones que proporciona CICS, como el asistente de servicios web y los manejadores de mensajes SOAP.

Si decide no utilizar SOAP, sus programas de aplicación serán responsables de analizar los mensajes de entrada y de construir los mensajes de salida.

¿Tiene pensado utilizar el asistente de servicios web de CICS para generar las correlaciones entre las estructuras de datos y los mensajes SOAP?

El asistente proporciona un despliegue rápido de muchas aplicaciones en una configuración de servicios web con poca programación o sin programación adicional. Cuando se necesita programación adicional, es sencillo, y se puede hacer sin cambiar la lógica empresarial existente.

Sin embargo, hay casos en los que es están mejor manejados sin utilizar el asistente de servicios web. Por ejemplo, si tiene un código existente que correlaciona estructuras de datos con mensajes SOAP, no hay ventaja a la hora de volver a organizar la aplicación con el asistente de servicios web.

Aunque el asistente de servicios web de CICS soporta los tipos de datos y estructuras más comunes, hay algunas que no se soportan. En esta situación, debe comprobar la lista de tipos de datos y estructuras no soportados para el lenguaje en cuestión, y tener en cuenta proporcionar una capa de programa que correlaciona los datos de aplicación a un formato que puede soportar el asistente. Si no es posible, deberá analizar el

mensaje. Para obtener detalles sobre qué puede y qué no puede soportar el asistente, consulte High-level language and XML schema mapping.

Si decide no utilizar el asistente de servicios web de CICS, puede utilizar una herramienta como IBM Developer for z Systems para crear los artefactos necesarios, y puede proporcionar su propio código para analizar los mensajes de entrada y construir mensajes de salida. También puede utilizar la API de interfaz de proveedor proporcionada.

¿Tiene previsto utilizar una descripción de servicio existente o crear una?

En algunas situaciones, se verá obligado a utilizar una descripción de servicio existente como punto de partida. Por ejemplo:

- La aplicación es un solicitante de servicio y está diseñada para invocar un servicio web existente.
- Su aplicación es un proveedor de servicios y desea ajustarse a la descripción de servicio estándar del sector existente.

En otras situaciones, es posible que necesite crear una descripción de servicio para la aplicación.

Qué hacer a continuación

Servicios web CICS y JSON

Existen diferentes formas de comenzar con los servicios web JSON en CICS. La forma más adecuada para usted depende de cuánto sepa ya y lo avanzados que sean sus planes para utilizar los servicios web.

Acerca de esta tarea

Una de las opciones para servicios web JSON implica el uso de z/OS Connect. Consulte *Getting started with z/OS Connect* para obtener más información sobre esta opción. A continuación se enumeran algunos puntos de partida para los servicios web JSON en CICS:

Procedimiento

- Instale la aplicación de ejemplo. CICS proporciona un ejemplo de una aplicación de gestión de catálogos, que se puede habilitar como un proveedor de servicios web JSON. Para hacerlo, utilice DFHLS2JS para generar un servicio web a partir de las estructuras de lenguaje proporcionadas. Podría utilizar un navegador web o una aplicación cliente de terceros para probar el servicio web JSON. Para obtener más información, consulte el apartado *Creating a service provider application from a data structure*.

Utilice la aplicación de ejemplo si desea una forma práctica de saber más sobre los servicios web en CICS. La aplicación de ejemplo se describe en *La aplicación de ejemplo del gestor de catálogos de CICS*.

- Planifique el despliegue de una aplicación como un proveedor de servicios. Es posible que ya sepa lo suficiente acerca de cómo va a utilizar los servicios web en CICS para comenzar a planificar sus aplicaciones y la infraestructura relacionada.
- Si desea ver un ejemplo de cómo podría utilizar los servicios web JSON, observe el vídeo *Take your transactions mobile in CICS Showcase*. Otra opción para los servicios web JSON implica el uso de z/OS Connect. Consulte *Getting started with z/OS Connect* para obtener más información sobre esta opción.

Conceptos de los servicios web JSON

Lea este tema para comprender los conceptos detrás de los servicios web JSON.

Servicios Web

Un “servicio web” es un término genérico para una función de software alojada en una ubicación direccionable mediante red. En un sentido general, podría implicar un servicio basado en cloud, un servicio de programa de utilidad o incluso una aplicación departamental. El término “servicio web” se puede utilizar también de forma más específica, como un servicio alojado que utiliza el protocolo SOAP y se describe utilizando un documento WSDL. Este sentido, más específico, es el que suele implicar el término “servicios web en CICS”. De todas formas, el término más general se utiliza habitualmente en la comunidad JSON cuando se describen servicios basados en JSON. Los servicios web JSON utilizan el término en su sentido genérico.

Hay algunas diferencias importantes entre SOAP y JSON:

- Un mensaje SOAP contiene datos XML, mientras que un mensaje JSON contiene datos JSON. JSON y XML son distintos mecanismos de codificación para describir datos estructurados. JSON tiende a ser un mecanismo de codificación más eficaz, por lo que un mensaje JSON habitual será más pequeño que el mensaje XML equivalente.
- JSON resulta fácil de integrar en aplicaciones JavaScript, no así XML. Esto hace que JSON sea el formato de datos preferido para muchos desarrolladores de aplicaciones móviles.
- SOAP proporciona un mecanismo para añadir Cabeceras a un mensaje y una gama de especificaciones para la calidad de servicio (como configuración de seguridad y transacciones distribuidas). JSON no proporciona este mecanismo, si no que se basa en los servicios del protocolo de red HTTP subyacente. Esto deriva en que haya menos opciones para proteger y configurar una carga de trabajo.
- Los servicios web SOAP se describen utilizando documentos WSDL. Los servicios web JSON tienen una estructura menos formal, no suelen tener conexión directa y prefieren documentación por ejemplo.
- Los servicios web SOAP tienen un formato de error explícito que implica mensajes de error SOAP. No existe ningún equivalente para JSON.

También hay muchas similitudes entre JSON y SOAP:

- La implementación CICS de JSON se deriva de la arquitectura SOAP y comparte muchos conceptos y artefactos.
- Ambos incluyen programas de utilidad fuera de línea que contribuyen en la correlación de datos de aplicación con respecto a la representación de datos externos. En el caso de SOAP, DFHLS2WS y DFHWS2LS, en el caso de JSON, DFHLS2JS y DFHJS2LS.
- El mecanismo de despliegue de ambas tecnologías implica un recurso PIPELINE, un recurso WEBSERVICE y un recurso URIMAP.

esquema JSON

Una de las desventajas de JSON en comparación con el protocolo SOAP es la dificultad a la hora de documentar la estructura de una interfaz de JSON. Los servicios web SOAP tienen la ventaja de los documentos WSDL, además de los

esquemas XML. Es posible que un documento WSDL no resulte sencillo de entender, pero hay muchas herramientas disponibles para trabajar con documentos WSDL.

El equivalente más próximo a JSON es la especificación de esquema JSON disponible en <http://json-schema.org/>. En el momento en el que se redacta, esto es un borrador de la especificación que está abriéndose cambio hacia el proceso de estandarización IETF. El asistente de JSON de CICS (DFHLS2JS y DFHJS2LS) proporciona una implementación parcial del borrador 4 de esta especificación emergente. DFHLS2JS se puede utilizar para generar un esquema JSON, y DFHJS2LS se puede utilizar para procesarlos.

Puede utilizar el esquema JSON para comprender la sintaxis válida y el modelo de contenido de un servicio web JSON implementado en CICS. La especificación del esquema JSON no tiene el mismo ecosistema de herramientas que la especificación del esquema XML, pero se espera que surja una nueva generación de herramientas JSON para utilizar este formato de datos.

Implementación de CICS de los servicios web basados en JSON

CICS es compatible con dos modalidades de servicio web JSON, solicitud/respuestas y RESTful. CICS admite también un escenario de programación en el que las aplicaciones puedan transformar los datos JSON a formatos de datos de estilo COBOL, o de dichos formatos en datos JSON.

Solicitud-respuesta

El patrón JSON de solicitud/respuestas es muy parecido al de los servicios web basados en SOAP de CICS. El servicio web se implemente utilizando un PROGRAM en CICS. El PROGRAM tiene formatos de datos de entrada y salida, que se describen utilizando estructuras de lenguaje (como libros de copias COBOL) y CICS es el responsable de transformar los mensajes JSON entrantes en datos de aplicación y de crear enlaces con la aplicación. La aplicación devuelve los datos de salida a CICS y CICS los transforma en datos JSON para devolverlos al cliente.

En este escenario, el cliente JSON se tiene que conectar a CICS utilizando el método HTTP POST.

Se puede desarrollar un servicio web JSON en modalidad de solicitud/respuestas tanto en modalidad de abajo a arriba como de arriba a abajo. En modalidad de abajo a arriba, se expone un PROGRAM existente de CICS como servicio web JSON utilizando el asistente de JSON DFHLS2JS. En la modalidad de arriba a abajo, se puede desarrollar un nuevo servicio web JSON para implementar una interfaz descrita utilizando esquemas JSON existentes. En modalidad de arriba a abajo, se utiliza el asistente de JSON DFHJS2LS para generar nuevas estructuras de lenguaje y es necesario crear una aplicación para utilizarlas.

El patrón solicitud-respuesta puede utilizarse para compilar servicios web JSON que tienen como destino COMMAREA o CICS PROGRAMs conectados por canal. Un servicio web JSON de solicitud/respuestas solo se puede utilizar en modalidad de proveedor (donde CICS actúa como servidor).

RESTful

Este escenario es distinto al de los servicios web SOAP. El concepto de un servicio web JSON de RESTful se describe con más detalle en el apartado

Conceptos de servicios web JSON de RESTful. Un servicio web JSON de RESTful implementa los principios de arquitectura del patrón de diseño de REpresentational State Transfer (REST). Es poco probable que este patrón de diseño resulte relevante para aplicaciones CICS existentes, por lo que solo está disponible en modalidad de arriba a abajo.

DFHJS2LS puede procesar un esquema JSON en modalidad RESTful. Es necesario escribir una aplicación para implementar el servicio y esta tendrá que comportarse de distintas maneras según el método HTTP que se utilizara para la solicitud entrante.

CICS implementa un estilo puro de aplicaciones RESTful, donde el formato de datos para POST (crear), GET (consultar) y PUT (sustituir) es el mismo.

Las aplicaciones de servicios web RESTful JSON deben utilizar un canal basado en la interfaz de programa; las COMMAREAs no se soportan. Un servicio web JSON de RESTful solo se puede utilizar en modalidad de proveedor (donde CICS actúa como servidor).

Modalidad de programación

En este escenario, una aplicación puede enlazarse con un programa proporcionado por CICS, DFHJSON, y pedirle que transforme los datos de aplicación en datos JSON, o datos JSON en datos de aplicación. Por ejemplo, una aplicación podría utilizar este recurso para generar datos JSON que enviar a un servicio web JSON remoto. Para hacerlo, debe ponerse en contacto con el servicio web JSON remoto mediante la API web de CICS.

CICS no dispone de soporte incorporado para servicios web JSON en modalidad de solicitante, pero una aplicación puede llamar a un servicio web JSON remoto mediante la modalidad de programación.

Conceptos de servicios web JSON de RESTful

Lea este tema para comprender los conceptos detrás de los servicios web RESTful.

Servicios web RESTful

REpresentational State Transfer, o REST, es un patrón de diseño para interactuar con los recursos almacenados en un servidor. Cada recurso tiene una identidad, un tipo de datos y da soporte a un conjunto de acciones.

El patrón de diseño RESTful se utiliza normalmente en combinación con HTTP, el lenguaje de Internet. En este contexto la identidad del recurso es su URI, el tipo de datos es su tipo de medios y las acciones se componen de los métodos HTTP estándar (GET, PUT, POST y DELETE).

Este estilo de servicio difiere de los servicios web de estilo solicitud/respuestas:

- Los servicios de solicitud/respuestas inician la interacción con una aplicación, mientras que los servicios RESTful generalmente interaccionan con datos (denominados 'recursos').
- Los servicios de solicitud/respuestas implican 'operaciones' definidas por la aplicación, pero los servicios RESTful evitan conceptos específicos de aplicaciones.
- Los servicios de solicitud/respuestas tienen distintos formatos de datos para cada mensaje, mientras que los servicios RESTful suelen compartir un formato de datos en distintos métodos HTTP.

Los cuatro métodos HTTP principales definen las cuatro operaciones que suelen implementar los servicios RESTful. El método HTTP POST se utiliza para crear un recurso, GET se utiliza para consultarlo, PUT se usa para modificarlo y DELETE sirve para eliminarlo. La arquitectura RESTful más habitual implica un modelo de datos compartido que se utiliza en las cuatro operaciones. Este modelo de datos define la entrada del método POST (crear), la salida del método GET (consultar) y la entrada del método PUT (sustituir). Este sencillo patrón de diseño es conocido en la comunidad de RESTful, pero no es el único patrón de diseño de RESTful. El código de estado HTTP se utiliza para indicar éxito o fracaso de la operación. Algunas API de RESTful se han diseñado de otra forma.

En ocasiones, los servicios web RESTful admiten un quinto método HTTP, denominado 'HEAD'. Este método es equivalente a GET, excepto que solo devuelve cabeceras HTTP y no datos del cuerpo. En ocasiones, se utiliza para probar la existencia de un recurso. No todas las API de RESTful admiten el uso del método HEAD.

Es poco probable que las aplicaciones CICS tradicionales coincidan con el patrón de arquitectura de RESTful. Las aplicaciones CICS típicas implementan varias operaciones y cada una de ellas tiene modelos de datos para los formatos de entrada y salida. No es probable que estas operaciones existentes se correlacionen directamente con los cuatro métodos HTTP. Es por ello que el patrón de arquitectura de RESTful está orientado sobre todo a las nuevas aplicaciones de CICS. Para exponer las aplicaciones CICS existentes como servicios RESTful es posible que tenga que acomodarlas a una nueva interfaz que se ajuste a los principios de RESTful.

El URI

La identidad de un servicio RESTful se indica mediante su URI. A URI puede estar formado por varios componentes, como el nombre de host, el número de puerto, la ruta y la cadena de consulta opcional. El nombre de dominio y el número de puerto se dirigen juntos a un recurso TCPIPService en CICS. Para obtener más información, consulte el apartado TCPIPService resources. La vía de acceso del URI es un calificador y puede ser suficiente para identificar de manera exclusiva el servicio. Sin embargo, muchos servicios RESTful utilizan una cadena de consulta adicional para identificar el recurso preciso. Tenga en cuenta los ejemplos siguientes:

- `http://www.example.org:10000/JSONServices/AccountService`
- `https://www.example.org:10000/JSONServices?Service=Account`

En el primer ejemplo, la vía de acceso del URI es `JSONServices/AccountService`. En el segundo ejemplo la vía de acceso es `JSONServices` y hay una cadena de consulta adicional `Service=Account`. Ambos estilos de URI se consideran aceptables para JSON. Esto es una diferencia importante en comparación con SOAP. En SOAP, se prefiere el primer estilo de URI.

CICS utiliza un recurso URIMAP para identificar el `WEBSERVICE` y `PIPELINE` que se utilizará al procesar un mensaje de entrada. URIMAP admite el uso de una serie de consulta como parte del atributo `path`. Por lo tanto, el URIMAP es adecuado para utilizarlo con ambos tipos de URI.

Planificación para utilizar servicios web JSON

Antes de planificar el uso de los servicios web JSON en CICS, necesita tener en cuenta estas preguntas para cada aplicación.

Antes de empezar

¿Tiene previsto utilizar sus programas de aplicación existentes o escribir otros nuevos?

Si sus aplicaciones existentes se han diseñado con una interfaz bien definida en cuanto a la lógica empresarial, probablemente podrá utilizarlas en una configuración de servicios web, como proveedor de servicios o como solicitante de servicio. Sin embargo, en la mayoría de los casos, tendrá que escribir un programa derivador que conecte la lógica empresarial con la lógica de servicios.

Si tiene previsto escribir aplicaciones nuevas, intente mantener la lógica empresarial separada de la lógica de servicios web y, una vez más, tendrá que escribir un programa derivador para proporcionar esta separación. Sin embargo, si la aplicación está diseñada pensando en los servicios web, resultará más sencillo escribir el programa derivador.

¿Tiene pensado utilizar el asistente de CICS para generar las correlaciones entre las estructuras de datos y esquemas JSON?

El asistente proporciona un despliegue rápido de muchas aplicaciones en una configuración de servicios web JSON con poca programación o sin programación adicional. Cuando se necesita programación adicional, es sencillo, y se puede hacer sin cambiar la lógica empresarial existente.

Sin embargo, hay casos en los que es están mejor manejados sin utilizar el asistente de JSON. Por ejemplo, si tiene un código existente que correlaciona estructuras de datos a mensajes JSON, no hay ventaja a la hora de volver a organizar la aplicación con el asistente de JSON.

Aunque el asistente de CICS soporta los tipos de datos y estructuras más comunes, hay algunas que no se soportan. En esta situación, debe comprobar la lista de tipos de datos y estructuras no soportados para el lenguaje en cuestión, y tener en cuenta proporcionar una capa de programa que correlaciona los datos de aplicación a un formato que puede soportar el asistente. Si no es posible, deberá analizar el mensaje. Para obtener detalles sobre qué puede y qué no puede soportar el asistente, consulte High-level language and JSON schema mapping.

Planificación de una aplicación de proveedor de servicios JSON

En general, las aplicaciones de CICS debe estructurarse para garantizar la separación de la lógica empresarial y la lógica de las comunicaciones. Seguir esta práctica le ayudará a desplegar aplicaciones existentes y nuevas en un proveedor de servicios web de forma sencilla. En algunas situaciones, necesitará interponer un sencillo programa derivador entre el programa de aplicación y el soporte de servicio web de CICS.

Figura 12 en la página 34 muestra una aplicación típica que se particiona para garantizar una separación entre la lógica de comunicación y la lógica empresarial.

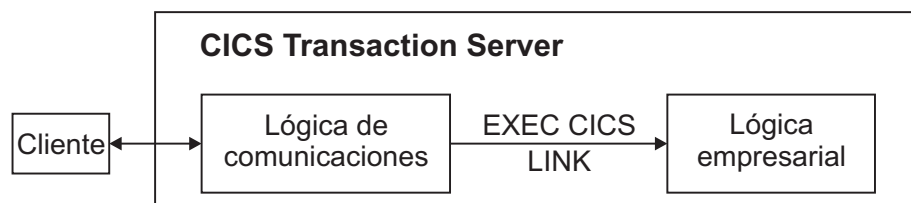


Figura 12. Aplicación particionada en lógica de comunicaciones y empresarial

En muchos casos, puede desplegar la lógica empresarial directamente como una aplicación de proveedor de servicios. Esto se ilustra en la Figura 13.

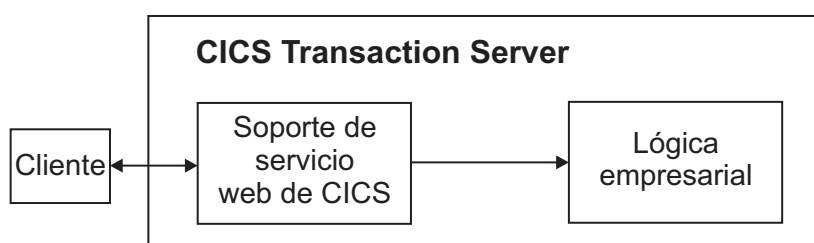


Figura 13. Despliegue simple de una aplicación CICS como proveedor de servicios web

Para utilizar este modelo simple, deben cumplirse las siguientes condiciones:

Si está utilizando el asistente de CICS para generar la correlación entre el esquema JSON y las estructuras de datos de aplicación:

El asistente de CICS debe soportar los tipos de datos utilizados en la interfaz para la lógica empresarial. Si no es este el caso, debe interponer un programa derivador entre el soporte web de CICS y la lógica empresarial.

También necesitará un programa derivador cuando despliegue un programa existente para proporcionar un servicio que se ajusta a una descripción de servicio web existente: si procesa la descripción de servicio web utilizando el asistente, es poco probable que las estructuras de datos resultantes coincidan con la interfaz para su lógica empresarial.

Cuando no está utilizando el asistente de CICS:

Los manejadores de mensajes de la interconexión de proveedor de servicios debe interactuar directamente con la lógica empresarial.

Uso de un programa derivador

Utilice un programa derivador cuando el asistente de CICS no pueda generar código para interactuar directamente con la lógica empresarial. Por ejemplo, la interfaz para la lógica empresarial puede utilizar una estructura de datos que el asistente de CICS no puede correlacionar directamente en un mensaje JSON. En esta situación, puede utilizar un programa derivador para proporcionar la manipulación de datos adicional que se necesita:

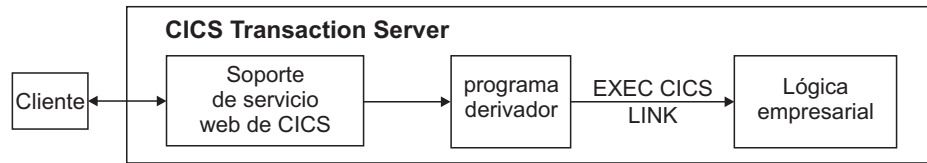


Figura 14. Despliegue de una aplicación CICS como proveedor de servicios web mediante un programa derivador

Deberá diseñar una segunda estructura de datos que pueda soportar el asistente y utilizarlas como la interfaz para el programa derivador. El programa derivador tiene entonces dos funciones simples para realizar:

- mover datos entre las dos estructuras
- invocar la lógica empresarial utilizando su interfaz existente

Manejo de errores

Si está pensando utilizar el asistente de CICS, también debe tener en cuenta cómo manejar los cambios de reversión cuando se produzcan errores. Cuando se recibe un mensaje de solicitud JSON desde un solicitante de servicio, CICS transforma el mensaje JSON justo antes de que pase al programa de aplicación. Si se produce un error durante esta transformación, CICS no retrotrae automáticamente ningún trabajo que se haya llevado a cabo en el mensaje. Por ejemplo, si piensa añadir proceso adicional en el mensaje JSON utilizando manejadores en la interconexión, debe decidir si se debe retrotraer algún cambio recuperable que ya hayan realizado.

En los mensajes JSON de salida, por ejemplo cuando el programa de aplicación de proveedor de servicios está enviando un mensaje de respuesta a un solicitante de servicio, si CICS se encuentra un error al generar el mensaje JSON de respuesta, todos los cambios recuperables realizados por el programa de aplicación se restituyen completamente. Debe tener en cuenta si añadir puntos de sincronización es adecuado para el programa de aplicación.

Planificación de una aplicación de solicitante de servicio JSON

CICS no proporciona soporte incorporado para los servicios web JSON modalidad solicitante. Si desea llamar a otros servicios web JSON desde su aplicación CICS, debe utilizar la interfaz enlazable y los mandatos de API **EXEC CICS WEB**.

Para obtener más información sobre cómo escribir su propio solicitante de servicio utilizando la interfaz enlazable, consulte Creación de una aplicación de cliente de servicio web JSON.

CICS y z/OS Connect

z/OS Connect habilita programas CICS para que se invoque con una interfaz de JavaScript Object Notation (JSON). z/OS Connect for CICS 1.0 se introdujo por primera vez como alternativa a las prestaciones JSON de Java Pipelines for JSON que se proporcionaban en CICS TS Feature Pack for Mobile Extensions y se integraban en CICS TS Versión 5.2. Desde entonces, z/OS Connect ha crecido en un producto aparte llamado z/OS Connect Enterprise Edition, con prestaciones adicionales. Esta sección describe las diferencias entre las ofertas alternativas y los pasos hacia la implementación.

Para obtener una descripción general de los servicios JSON, consulte "CICS as a service provider for JSON requests en IBM Knowledge Center.

Consideraciones de mantenimiento para z/OS Connect Enterprise Edition

Como z/OS Connect Enterprise Edition puede ejecutarse de forma autónoma o en Liberty incluido en CICS, los dos entornos deben describirse de forma exclusiva. Puede ejecutar ambos entornos de forma simultánea, pero no son mutuamente excluyentes, por lo que es necesario tener cuidado cuando se aplica mantenimiento a las bibliotecas de cliente de WebSphere Liberty Profile (WLP) que contienen módulos BBOA*.

Para obtener más información, consulte **Tener el mantenimiento de CICS TS 5.3 y z/OS Connect EE 2.0 en sincronización.**

z/OS Connect Enterprise Edition

z/OS Connect Enterprise Edition es un producto de IBM que se puede solicitar de forma individual. Añade las prestaciones de z/OS Connect for CICS 1.0, que incluyen soporte para los servicios JSON. z/OS Connect Enterprise Edition habilita los desarrolladores de API para construir API de JSON desde servicios JSON. Las API se construyen y empaquetan con el Editor de API basado en Eclipse que se proporciona con z/OS Connect Enterprise Edition, después, se despliega en el tiempo de ejecución de z/OS Connect. El paquete de API incluye definiciones de Swagger 2.0 para facilitar a los desarrolladores la incorporación de las API en sus aplicaciones. Las prestaciones de z/OS Connect clave, como la comprobación de seguridad de autorización para la invocación de servicios, la creación de registros de System Management Facility (SMF) y el registro de solicitudes de servicio RESTful también se aplican a las APIs.

z/OS Connect Enterprise Edition se puede configurar para que se ejecute en un servidor Liberty en CICS. Esta configuración habilita una conectividad de alto rendimiento local para programas CICS.

Los servicios JSON que se desplegaron para su uso con z/OS Connect for CICS 1.0 y la mayoría de los servicios JSON que se desplegaron para su uso con Java Pipelines for JSON, se pueden alojar en z/OS Connect Enterprise Edition. Consulte "z/OS Connect for CICS 1.0" para obtener más información. Sin embargo, los usuarios del Editor de API proporcionado con z/OS Connect Enterprise Edition deben tener en cuenta algunas restricciones. Consulte Using APIs from z/OS Connect Enterprise Edition para obtener más información.

Para obtener más información sobre z/OS Connect Enterprise Edition, consulte el apartado z/OS Connect Enterprise Edition documentation in IBM Knowledge Center.

z/OS Connect for CICS 1.0

z/OS Connect for CICS 1.0 es una característica gratuita proporcionada en CICS TS de la versión 5.2. Es en general equivalente a Java Pipelines for JSON y la mayoría de los servicios web JSON pueden volver a desplegarse de un entorno a otro sin cargos a las aplicaciones o a archivos WSBIND. Sin embargo, el URI y la configuración de seguridad pueden ser diferentes en cada entorno. Los servicios web JSON desplegados en z/OS Connect for CICS 1.0 también se pueden desplegar en z/OS Connect Enterprise Edition.

La versión de z/OS Connect incluida en CICS TS comparte mucho de la función de z/OS Connect en WebSphere Liberty for z/OS, pero está optimizada para el acceso local a CICS y utiliza servicios CICS estándar. Si está familiarizado con z/OS Connect en WebSphere Liberty for z/OS, las diferencias principales entre esto y z/OS Connect for CICS 1.0 son las siguientes:

- z/OS Connect for CICS 1.0 sólo soporta el despliegue de servicios web JSON para CICS. No se puede utilizar como destino para otros subsistemas z/OS como IMS o Batch.
- z/OS Connect for CICS 1.0 se integra con recursos CICS. Los servicios web JSON se despliegan utilizando recursos WEBSERVICE y URIMAP.
- Las técnicas de habilitación de servicio JSON de Java Pipelines for JSON se soportan con z/OS Connect for CICS 1.0, incluidas aquellas que no están normalmente disponibles en z/OS Connect.
- Los servicios JSON se pueden desarrollar utilizando estrategias de despliegue ascendentes y descendentes.
- Los servicios web JSON generados para Java Pipelines for JSON se pueden desplegar en z/OS Connect for CICS 1.0.

Para obtener más información sobre z/OS Connect en WebSphere Liberty for z/OS, consulte IBM z/OS Connect overview.

Interconexiones Java para JSON (o el paquete de características para Mobile Extensions)

Mobile Extensions Feature Pack 1.0 ha trabajado con CICS TS for z/OS versión 5.1 y versión 4.2. El paquete de características ha proporcionado la prestación para exponer las aplicaciones CICS como servicios web RESTful con cargas útiles JSON, invocar aplicaciones JSON existentes y convertir a JSON a y desde datos de aplicación. Desde la versión 5.2, esta prestación se ha integrado en CICS TS.

Comparación de prestaciones de z/OS Connect

Esta sección compara las prestaciones de z/OS Connect Enterprise Edition y z/OS Connect for CICS 1.0. También se incluye una comparación para Java Pipelines for JSON (o el paquete de características para Mobile Extensions) porque precedía z/OS Connect for CICS 1.0.

Tabla 1. Comparación de las prestaciones de z/OS Connect Enterprise Edition, z/OS Connect for CICS 1.0 y Java Pipelines for JSON (o del paquete de características para Mobile Extensions)

Prestación	z/OS Connect Enterprise Edition	z/OS Connect for CICS 1.0	Java Pipelines for JSON (o el paquete de características para Mobile Extensions)
Soporte para las prestaciones de z/OS Connect estándar	Sí. El servicio de descubrimiento requiere configuración adicional.	Sí. El servicio de descubrimiento requiere configuración adicional.	No
Habilitar programas CICS como servicios web JSON	Sí	Sí	Sí

Tabla 1. Comparación de las prestaciones de z/OS Connect Enterprise Edition, z/OS Connect for CICS 1.0 y Java Pipelines for JSON (o del paquete de características para Mobile Extensions) (continuación)

Prestación	z/OS Connect Enterprise Edition	z/OS Connect for CICS 1.0	Java Pipelines for JSON (o el paquete de características para Mobile Extensions)
Soporte para servicios JSON RESTful	Sí	Sí	Sí
Soporte para API de RESTful	Sí	No	No
Soporte para iniciar servicios JSON remotos	No. Utilice DFHJSON.	No. Utilice DFHJSON.	No (utilice DFHJSON)
Soporte para la transformación de datos de CICS que utiliza asistentes DFHLS2JS y DFHJ2LS	Para servicios, sí. Para APIs, utilice los asistentes proporcionados por z/OS Connect EE	Sí	Sí
Soporte para especificación de esquema JSON	Sí (borrador-04)	Sí (borrador-04)	Sí (borrador-04)
Opción de tecnologías de analizador para la transformación JSON	Sí	Sí	No
Soporte para programas de manejador PIPELINE	No	No	Sí
Soporte para programas de interceptor	Sí, utilizando interceptores de z/OS Connect	Sí, utilizando interceptores de z/OS Connect	Sí, utilizando manejadores de Axis2
Acceso a las aplicaciones para contenedores de control CICS	Sí	Sí	Sí
Despliegue de WSBIND a través de un recurso CICS PIPELINE	Para servicios, sí. El despliegue de API implica artefactos adicionales (consulte la documentación de z/OS Connect EE en IBM Knowledge Center para obtener detalles).	Sí	Sí
Configuración con los recursos URIMAP y WEBSERVICE	Para servicios, sí.	Sí	Sí
Configuración de red	Utilizando WebSphere Liberty	Utilizando WebSphere Liberty	Utilizando un recurso TCPIP SERVICE

Tabla 1. Comparación de las prestaciones de z/OS Connect Enterprise Edition, z/OS Connect for CICS 1.0 y Java Pipelines for JSON (o del paquete de características para Mobile Extensions) (continuación)

Prestación	z/OS Connect Enterprise Edition	z/OS Connect for CICS 1.0	Java Pipelines for JSON (o el paquete de características para Mobile Extensions)
Configuración de seguridad	Utilizando la seguridad de WebSphere Liberty y CICS	Utilizando la seguridad de WebSphere Liberty y CICS	Utilizando CICS
Soporte para la Seguridad de la capa de transporte (TLS)	Sí	Sí	Sí
Estadísticas, supervisión, salidas de usuario y otra infraestructura de diagnóstico	Coherente con WebSphere Liberty en CICS	Coherente con WebSphere Liberty en CICS	Coherente con interconexiones en CICS
Entorno de JVM	Un servidor Liberty JVM que está aislado preferiblemente para su uso por parte de z/OS Connect	Un servidor Liberty JVM que está aislado preferiblemente para su uso únicamente por parte de z/OS Connect	Un servidor que no es OSGi JVM configurado con JAVA_PIPELINE=YES , aislado preferiblemente para su uso únicamente por parte de Java Pipelines for JSON

Capítulo 2. Configuración de servicios web en CICS

Puede configurar CICS para dar soporte a servicios web, donde las aplicaciones CICS puede convertirse en solicitantes de servicio web o proveedores de servicios. CICS soporta diferentes especificaciones de servicio web, incluidos los archivos adjuntos binarios y el direccionamiento de servicios web. También puede configurar CICS para aceptar solicitudes de servicios web de WebSphere MQ o HTTP, y recuperar archivos WSDL desde WSRR.

Configuración del sistema CICS para servicios web

Antes de que pueda utilizar los servicios web, el sistema CICS debe estar configurado correctamente.

Procedimiento

1. Asegúrese de que ha instalado el soporte Language Environment para PL/I. Para obtener más información, consulte Instalación del soporte Language Environment.
2. Active el soporte z/OS para Unicode. Debe habilitar los servicios de conversión de z/OS e instalar una imagen de conversión que especifica las conversiones de datos que desea que CICS lleve a cabo entre mensajes SOAP y un programa de aplicación. Para obtener más información, consulte el apartado z/OS Unicode Services User's Guide and Reference.

Recursos de CICS para servicios web

Los recursos PIPELINE, WEBSERVICE, URIMAP y TCPIPService soportan servicios web en CICS.

PIPELINE

Se necesita una definición de recursos PIPELINE para cada servicio web. Ofrece información sobre los programas de gestión de mensajes que actúan en una solicitud de servicios y en la respuesta. Por lo general, una única definición de recursos PIPELINE define una infraestructura que pueden utilizar muchas aplicaciones. La información sobre los manejadores de mensajes se proporciona indirectamente: la definición de recurso PIPELINE especifica el nombre de un archivo z/OS UNIX que contiene una descripción XML de manejadores y su configuración.

Un recurso PIPELINE creado para un solicitante de servicio no se puede utilizar para un proveedor de servicios y viceversa. Los dos tipos de definición PIPELINE se distinguen por el contenido del archivo de configuración de interconexión que se especifica en el atributo CONFIGFILE: para un proveedor de servicios, el elemento de nivel superior es <provider_pipeline>; para un solicitante de servicio, es <requester_pipeline>.

WEBSERVICE

Una definición de recurso WEBSERVICE sólo se necesita cuando la correlación entre la estructura de datos de aplicación y los mensajes SOAP se ha generado utilizando el asistente de servicios web CICS. Define aspectos de un entorno de ejecución para un programa de aplicación CICS desplegado en un entorno de servicios web.

Aunque CICS proporciona los mecanismos de definición de recursos usuales para los recursos WEBSERVICE, se suelen crear automáticamente desde un archivo de enlace de servicio web cuando se explora el directorio de recogida para la definición de recurso PIPELINE. Esto puede ocurrir cuando el recurso PIPELINE está instalado o como resultado de un mandato PERFORM PIPELINE SCAN. Los atributos aplicados al recurso WEBSERVICE en este caso provienen de un archivo de enlace de servicios web, que crea el asistente de servicios web; la información del archivo de enlace viene de la descripción de servicio web, o se proporciona como parámetro del asistente de servicios web.

Un recurso WEBSERVICE creado para un solicitante de servicio no se puede utilizar para un proveedor de servicios y viceversa. Los dos tipos de recurso WEBSERVICE se distinguen por el atributo PROGRAM en la definición de recurso: para un proveedor de servicios, se debe especificar el atributo; para un solicitante de servicio, se debe omitir.

URIMAP

Se necesita una definición URIMAP en un proveedor de servicio cuando contiene información que correlaciona el URI de una solicitud de servicio web de entrada con los demás recursos (como el recurso PIPELINE) que dará servicio a la solicitud. Esta definición URIMAP también es necesaria si está utilizando la autenticación básica HTTP, porque la definición de recursos URIMAP especifica que la información de ID de usuario de solicitante de servicio se pasa en una cabecera de autorización HTTP al proveedor de servicios.

Puede existir una segunda definición URIMAP opcional en el proveedor de servicios para el descubrimiento WSDL. Esta definición de recursos URIMAP contiene información que correlaciona el URI de una solicitud de entrada para el documento WSDL o documentos asociados al servicio web.

Para proveedores de servicios desplegados utilizando el asistente de servicios web CICS, aunque CICS proporciona los mecanismos de definición de recursos habituales, los recursos URIMAP normalmente se crean de forma automática cuando se explora el directorio de recogida. Esta exploración se produce cuando el recurso PIPELINE está instalado o como resultado de un mandato PERFORM PIPELINE SCAN. El recurso URIMAP que proporciona información a CICS para asociar el recurso WEBSERVICE a un URI específico es un recurso necesario. Los atributos para este recurso se especifican por un archivo de enlace de servicio web en el directorio de recogida. El recurso URIMAP que proporciona información a CICS para asociar el archivo de archivado WSDL o el documento WSDL a un URI específico es un recurso opcional y se crea si hay presente un archivo WSDL o archivo de archivado WSDL en el directorio de recogida. Para obtener información sobre la creación de recursos URIMAP para proveedores de servicios web, consulte *Creating a web service provider by using the web services assistant*.

Para solicitantes de servicio, CICS no cree recursos URIMAP de forma automática cuando el recurso PIPELINE esté instalado o como resultado de un mandato PERFORM PIPELINE SCAN. Los solicitantes de servicio no son necesarios para utilizar recursos URIMAP cuando hacen solicitudes; pueden especificar el URI de la solicitud de salida directamente en el programa de aplicación. Sin embargo, si crea un recurso URIMAP para la solicitud de cliente, y los solicitantes de servicio utilizan el recurso URIMAP para proporcionar el URI, obtendrá estas ventajas:

- Los administradores del sistema puede gestionar todos los cambios en el punto final de la conexión, por lo que no es necesario volver a compilar sus aplicaciones si cambia el URI de un proveedor de servicios.
- Tiene la opción de hacer que CICS mantenga abiertas las conexiones que se abrieron con el recurso URIMAP después de utilizarlas, y colocarlas en una agrupación para que la aplicación las vuelva a utilizar en solicitudes posteriores, o para que las vuelva a utilizar otra aplicación que llame al mismo servicio. La técnica agrupación de conexiones solo está disponible si se especifica un recurso URIMAP que tenga el conjunto de atributos SOCKETCLOSE. Para obtener más información acerca de los beneficios de rendimiento de la técnica de agrupación de conexiones, consulte el apartado Connection pooling for HTTP client performance.

La configuración de atributos de recurso URIMAP de una determinada manera puede habilitar las solicitudes de entrada que se están procesando mediante transacciones de usuario conectadas directamente e ignorando la tarea de conexión web. Para obtener más información, consulte el apartado HTTP requests are processed by directly attached user transactions.

TCPIPSERVICE

Se necesita una definición TCPIPSERVICE en el proveedor de servicios que utilice el transporte HTTP. Contiene información sobre el puerto donde se reciben las solicitudes de entrada.

Los recursos necesarios para soportar un programa de aplicación particular dependen de los siguientes criterios:

- Si el programa de aplicación es un proveedor de servicios o un solicitante de servicio.
- Si la aplicación se despliega con el asistente de servicios web CICS.

Proveedor o solicitante de servicio	Asistente de servicios web de CICS utilizado	PIPELINE necesario	WEBSERVICE necesario	URIMAP necesario	TCPIPSERVICE necesario
Proveedor	Sí	Sí	Sí (pero consulte la nota 1)	Sí (pero consulte la nota 1)	Consulte la nota 2
Proveedor	No	Sí	No	Sí	Consulte la nota 2
Solicitante	Sí	Sí	Sí	Consulte la nota 3	No
Solicitante	No	Sí	No	3	No

Notas:

1. Cuando se utiliza el asistente de servicios web CICS para desplegar un programa de aplicación, se pueden crear automáticamente un recurso WEBSERVICE y dos recursos URIMAP cuando se explora el directorio de recogida de PIPELINE. El primer recurso URIMAP es necesario y proporciona información a CICS para asociar el recurso WEBSERVICE al URI específico. El segundo recurso URIMAP es opcional y proporciona información a CICS para asociar el archivo de archivado WSDL o un documento WSDL a un URI específico así los solicitantes externos pueden utilizar el URI para descubrir el archivo de archivado WSDL o el documento WSDL. El directorio de recogida

de la exploración PIPELINE se produce cuando se instala el recurso PIPELINE o como resultado de un mandato PERFORM PIPELINE SCAN.

2. Se necesita un recurso TCPIPSERVICE cuando se utiliza el transporte HTTP. Cuando se utiliza el transporte WebSphere MQ, no es necesario un recurso TCPIPSERVICE.
3. El recurso URIMAP es opcional para un solicitante de servicio, y el asistente de servicios web de CICS no genera uno automáticamente. Cuando define sus propios recursos URIMAP para que los utilicen los solicitantes de servicio, puede implementar la técnica de agrupación de conexiones, y gestionar cambios para los URI para los proveedores de servicios.

La configuración de atributos de recurso TCPIP de una determinada manera puede habilitar las solicitudes de entrada que se están procesando mediante transacciones de usuario conectadas directamente e ignorando la tarea de conexión web. Para obtener más información, consulte el apartado HTTP requests are processed by directly attached user transactions.

Normalmente, cuando despliega muchas aplicaciones de servicios web en un sistema CICS, tiene más de una de cada tipo de recurso. En este caso, puede compartir algunos recursos entre aplicaciones. Cada recurso o archivo de servicios web está asociado a uno o más recursos CICS de otros tipos.

Tabla 2. Otros recursos de CICS asociados a cada uno de los recursos y archivos de servicios web

Recurso o archivo de servicios web	Recursos asociados
Archivo de configuración de interconexión	<ul style="list-style-type: none"> Más de un recurso PIPELINE que hace referencia al archivo.
PIPELINE	<ul style="list-style-type: none"> Más de un recurso URIMAP que hace referencia al recurso PIPELINE. Más de un recurso WEBSERVICE que hace referencia al recurso PIPELINE. Más de un archivo de enlace de servicio web en el directorio de recogida del recurso PIPELINE.
Archivo de enlace de servicio web	<ul style="list-style-type: none"> Un recurso URIMAP que se genera automáticamente desde el archivo de enlace. Puede definir más recursos URIMAP para un proveedor de servicios y puede definir recursos URIMAP para un solicitante de servicio. Un recurso WEBSERVICE que se genera automáticamente desde el archivo de enlace. Puede definir más recursos WEBSERVICE si lo necesita.
WEBSERVICE	<ul style="list-style-type: none"> Más de un recurso URIMAP. Si el recurso WEBSERVICE se genera automáticamente a partir del archivo de enlace para un proveedor de servicios, CICS genera un recurso URIMAP correspondiente. Puede definir más recursos URIMAP para un proveedor de servicios y puede definir recursos URIMAP para un solicitante de servicio.

Tabla 2. Otros recursos de CICS asociados a cada uno de los recursos y archivos de servicios web (continuación)

Recurso o archivo de servicios web	Recursos asociados
URIMAP	<ul style="list-style-type: none"> Sólo un recurso TCPIPService cuando se nombra explícitamente en el recurso URIMAP.
TCPIPService	<ul style="list-style-type: none"> Muchos recursos URIMAP.

Descubrimiento de servicios web

Los documentos WSDL asociados a un servicio web de modalidad de proveedor se publican automáticamente en la web.

Existe un convenio entre entornos de alojamiento de servicio web que permite que un cliente remoto consulte el WSDL para un servicio web (normalmente un desarrollador de aplicaciones que utiliza un navegador web) utilizando el URI para el servicio web que tiene como sufijo `?wsdl`. Este convenio puede facilitar la distribución de WSDL a partes interesadas sin la necesidad de un repositorio WSDL formal. Este convenio se implementa en CICS.

Por ejemplo, puede tener un servicio web alojado en CICS y publicado bajo el siguiente URI:

`http://www.example.org:1234/example/WebService`

El documento WSDL asociado se puede recuperar solicitando el siguiente URI mediante un navegador web:

`http://www.example.org:1234/example/WebService?wsdl`

Los documentos WSDL para el proveedor de servicios se pueden publicar para descubrimiento utilizando recursos URIMAP. Al instalar cada recurso PIPELINE, CICS explora el directorio especificado en el atributo WSDIR del recurso PIPELINE (directorio de recogida). Si este directorio contiene un archivo de archivado WSDL o un documento WSDL, se instalará un segundo recurso URIMAP. Este nuevo recurso URIMAP proporciona a CICS la información necesaria para asociar el archivo de archivado WSDL o un documento WSDL con un determinado URI para que los solicitantes externos puedan utilizar el URI para descubrir el archivo de archivado WSDL o el documento WSDL. Este URI tiene la misma vía de acceso que el URI asociado con WEBSERVICE con el sufijo `?wsdl` agregado.

El archivo de archivado WSDL puede incluir uno o varios documentos WSDL. Si el directorio de recogida contiene un archivo de archivado WSDL y un documento WSDL, el URI solo devolverá el archivo de archivado WSDL. El formato del archivo de archivado que se soporta es el tipo de archivo `.zip`. También se pueden descubrir los archivos de archivado WSDL o los documentos WSDL mediante SPI y CEMT. El documento WSDL en un archivo de archivado WSDL se puede utilizar para la validación de mensaje SOAP.

Configuración de CICS para utilizar el transporte WebSphere MQ

Para utilizar el transporte WebSphere MQ con servicios web SOAP en CICS, debe configurar la región CICS según corresponda.

Acerca de esta tarea

Nota: No puede utilizar el transporte Websphere MQ para servicios web JSON.

Procedimiento

1. Incluya la biblioteca WebSphere MQ *thlqual*.SCSQAUTH en la concatenación STEPLIB en el procedimiento CICS. Incluya la biblioteca después de las bibliotecas de CICS para asegurarse de que se está utilizando el código correcto. *thlqual* es el calificador de alto nivel para las bibliotecas de WebSphere MQ.
2. Incluya la siguiente biblioteca de WebSphere MQ en la concatenación DFHRPL en el procedimiento CICS. Incluya las bibliotecas después de las bibliotecas de CICS para asegurarse de que se utiliza el código correcto.

thlqual.SCSQCICS

thlqual.SCSQLOAD

thlqual.SCSQAUTH

thlqual es el calificador de alto nivel para las bibliotecas de WebSphere MQ. Si está utilizando la salida cruzada de API CICS-WebSphere MQ (CSQCAPX), también debe añadir el nombre de la biblioteca que contiene el módulo de carga para el programa. La biblioteca SCSQCICS sólo es necesaria si desea ejecutar los ejemplos proporcionados por WebSphere MQ. De lo contrario, se puede eliminar del procedimiento CICS.

3. Instale un recurso MQCONN para la región de CICS. El recurso MQCONN especifica los atributos de la conexión entre CICS y WebSphere MQ, incluido el nombre del gestor de colas WebSphere MQ predeterminado o el grupo de compartición de colas para la conexión. Para obtener más información, consulte el apartado Setting up an MQCONN resource.
4. Especifique el parámetro de inicialización del sistema CICS **MQCONN=YES** para iniciar la conexión de CICS-WebSphere MQ automáticamente durante la inicialización de CICS. Para obtener detalles, consulte MQCONN system initialization parameter.
5. Si está utilizando el adaptador de CICS-WebSphere MQ en un sistema CICS que tiene comunicación entre regiones (IRC) para sistemas CICS, remotos, asegúrese de que el recurso IRC está OPEN antes de iniciar el adaptador, especificando el parámetro de inicialización del sistema de CICS, IRCSTRT=YES. El recurso IRC debe estar OPEN si el método de acceso IRC se define como memoria cruzada; es decir, ACCESSMETHOD(XM).
6. Asegúrese de que los identificadores del conjunto de caracteres codificados (CCSIDs) utilizados por el gestor de colas y por CICS, y las páginas de códigos UTF-8 y UTF-16 están configuradas para los servicios de conversión z/OS. La página de códigos de CICS se especifica en el parámetro de inicialización del sistema **LOCALCCSID**.
7. Actualice los CSD de CICS de la siguiente manera:
 - a. Si no comparte el CSD con releases anteriores de CICS, elimine los grupos CSQCAT1 y CSQCKB, y cualquier copia de esos grupos o de los elementos de esos grupo del CSD. También debe suprimir CKQQ TDQUEUE del grupo CSQCAT1. La definición para CKQQ se proporciona ahora en el grupo CSD de CICS, DFHDCTG.
 - b. Si comparte el CSD con releases anteriores de CICS, asegúrese de que CSQCAT1 y CSQCKB, y cualquier copia de esos grupos o de su contenido, no están instalados para CICS TS 4.1 o CICS TS 3.2. También debe suprimir CKQQ TDQUEUE del grupo CSQCAT1. La definición para CKQQ se

proporciona ahora en el grupo CSD de CICS, DFHDCTG. Para releases de CICS TS anteriores a CICS TS 3.2, instale los grupos CSQCAT1 y CSQCKB como parte de una lista de grupos después de instalar DFHLIST, para sustituir el grupo DFHMQ e instalar correctamente las definiciones necesarias.

8. Actualice las definiciones de WebSphere MQ para la cola de mensajes no entregados, la cola de transmisión predeterminada y los objetos de adaptador de CICS-WebSphere MQ. Puede utilizar el CSQ4INYG de ejemplo, pero puede que necesite cambiar el nombre de cola de inicio para que coincida con el nombre de cola de inicio predeterminado en la definición de recurso MQINI para la región de CICS. Puede utilizar este miembro en la concatenación CSQINP2 DD del procedimiento de inicio del gestor de colas o puede utilizarlo como entrada a la función COMMAND del programa de utilidad CSQUTIL para emitir los mandatos DEFINE necesarios. Utilizar el programa de utilidad CSQUTIL es preferible porque no tiene que volver a definir estos objetos cada vez que reinicie WebSphere MQ.

El transporte WebSphere MQ

CICS puede recibir y enviar mensajes SOAP a WebSphere MQ utilizando el transporte WebSphere MQ, en el rol de proveedor de servicios y de solicitante de servicio.

Como **proveedor de servicios**, CICS utiliza el desencadenante WebSphere MQ para procesar mensajes SOAP desde una cola de aplicación. El desencadenante funciona utilizando una cola de inicio y colas locales. Una definición de cola local (aplicación) incluye la siguiente información:

- Los criterios para cuando se genera un mensaje de desencadenante. Por ejemplo cuando llega el primer mensaje a la cola local, o para cada mensaje que llega a la cola local. Para el proceso SOAP de CICS, especifique que el desencadenante tienen lugar cuando llega el primer mensaje a la cola local.
La definición de cola local también puede especificar que los datos de desencadenante se pasan a la aplicación de destino, y en caso de un proceso SOAP de CICS (CPIL de transacción), esto especifica la URL de destino predeterminada que se va a utilizar si esto no pasa con el mensaje de entrada.
- El *nombre de proceso* que identifica la *definición de proceso*. La definición de proceso describe cómo se procesa el mensaje. En el caso del proceso SOAP de CICS, especifique la transacción CPIL.
- El nombre de la cola de inicio que se debe enviar al mensaje de desencadenante.

Cuando llega un mensaje a la cola local, Queue Manager genera y envía un mensaje desencadenante a la cola de inicio especificada. El mensaje de desencadenante incluye la información de la definición de proceso. El supervisor desencadenante recupera el mensaje de desencadenante de la cola de inicio y planifica la transacción CPIL para iniciar el proceso de mensajes en la cola local. Para más información sobre el desencadenante, consulte .

Puede configurar CICS, para que cuando llegue un mensaje a la cola local, el supervisor de desencadenante (proporcionado por WebSphere MQ) planifique la transacción CPIL para procesar los mensajes en la cola local y dirigir la interconexión SOAP de CICS para procesar mensajes SOAP en la cola.

Cuando CICS construye una respuesta para un mensaje SOAP que se recibe desde WebSphere MQ, el campo ID de correlación se llena con el ID de mensaje del mensaje de entrada, a menos que se haya establecido la opción de informe

MQRO_PASS_CORREL_ID. Si se ha establecido esta opción de informe, el ID de correlación se propaga desde el mensaje de entrada a la respuesta.

Como **solicitante de servicio**, en solicitudes de salida, puede especificar que las respuestas para el servicio web de destino se devuelven en una cola de respuestas concreta.

En ambos casos, CICS y WebSphere MQ requieren configuración para definir las colas y recursos necesarios.

Definición de colas locales en un proveedor de servicios

Para utilizar el transporte WebSphere MQ en un proveedor de servicios, debe definir una o varias colas locales que almacenan mensajes de solicitud hasta que se procesan, y un proceso desencadenante que especifica la transacción CICS que procesará los mensajes de solicitud.

Procedimiento

1. Defina una cola de inicio. Utilice el mandato siguiente:

```
DEFINE  
QLOCAL('cola_inicio')  
DESCR('descripción')
```

donde *cola_inicio* es el mismo valor que el especificado para el atributo QNAME de la definición de recurso MQMONITOR instalada para la región de CICS, o el valor especificado para el atributo INITQNAME de la definición de recurso MQCONN instalada.

2. Para cada cola de solicitud local, defina un objeto QLOCAL. Utilice el mandato siguiente:

```
DEFINE  
QLOCAL('nombre_cola')  
DESCR('descripción')  
PROCESS(nombre_proceso)  
INITQ('cola_inicio')  
TRIGGER  
TRIGTYPE(FIRST)  
TRIGDATA('servicio_destino_predeterminado')  
BOTHRESH(nnn)  
BOQNAME('nombre_cola_reenviados')
```

donde:

- *nombre_cola* es el nombre de cola local.
- *nombre_proceso* es el nombre de la instancia de proceso que identifica la aplicación iniciada por el gestor de colas cuando se produce un suceso desencadenante. Especifique el mismo nombre en cada objeto QLOCAL.
- *cola_inicio* es el nombre de la cola de inicio que se va a utilizar; por ejemplo, la cola de inicio especificada en el atributo QNAME de la definición de recurso MQMONITOR instalada para la región de CICS.
- *servicio_destino_predeterminado* es el servicio de destino predeterminado que se va a utilizar si no se especifica un servicio en la solicitud. El servicio de destino tiene el formato '/string' y se utiliza para coincidir con la vía de acceso de una definición URIMAP; por ejemplo, '/SOAP/test/test1'. El primer carácter debe ser '/'.

- *nnn* es el número de reentradas que se intentan.
 - *nombre_cola_reenviados* es el nombre de la cola a la que se envían los mensajes que han fallado.
3. Defina un objeto PROCESS que especifica el proceso desencadenante. Utilice el mandato siguiente:

```
DEFINE
PROCESS(nombre_proceso)
APPLTYPE(CICS)
APPLICID(CPIL)
```

donde:

nombre_proceso es el nombre del proceso, y debe ser el mismo que el nombre que se utiliza al definir las colas de solicitud.

Definición de colas locales en un solicitante de servicio

Cuando utiliza el transporte WebSphere MQ para solicitudes de salida en un solicitante de servicio, puede especificar en el URI para el servicio web de destino que deben devolver sus respuestas en una cola de respuestas predefinida. Si lo hace, debe definir cada cola de respuestas con un objeto QLOCAL.

Acerca de esta tarea

Si el URI asociado a una solicitud no especifica una cola de respuestas, CICS utilizará una cola dinámica para la respuesta.

Procedimiento

Opcional: Para definir cada objeto QLOCAL que especifica una cola de respuestas predefinidas, utilice el mandato siguiente.

```
DEFINE
QLOCAL('cola_respuestas')
DESCR('descripción')
BOTHRESH(nnn)
```

donde:

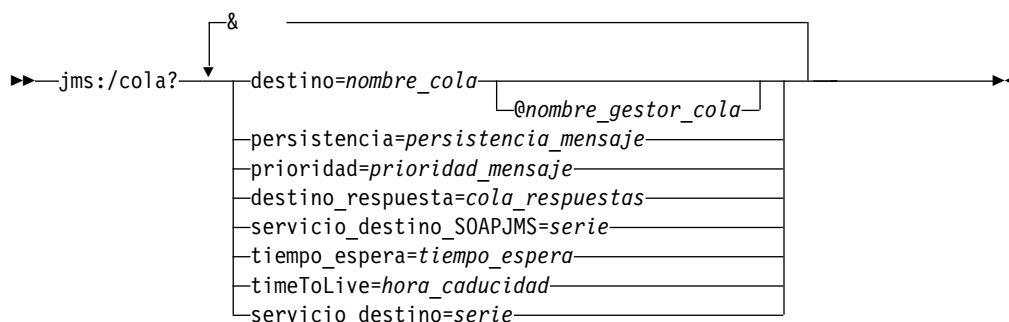
cola_respuestas es el nombre de cola local.

nnn es el número de reentradas que se intentarán.

El URI para el transporte IBM MQ

Cuando la comunicación entre el solicitante de servicio y el proveedor de servicios utiliza IBM MQ, el URI del destino está en un formato que identifica el destino como una cola e incluye información para especificar cómo IBM MQ debe gestionar la solicitud y la respuesta.

Sintaxis



Opciones

CICS utiliza las siguientes opciones; otros proveedores de servicio web pueden utilizar más opciones que no se describen aquí. El URI entero se pasa al proveedor de servicios, pero CICS ignora las opciones que no soporta y que están codificadas en el URI. CICS no distingue entre mayúsculas y minúsculas en los nombres de opción. Sin embargo, otras implicaciones que soportan este estilo de URI distinguen entre mayúsculas y minúsculas.

destino=nombre_cola [*@nombre_gestor_cola*]

nombre_cola es el nombre de la cola de entrada en el gestor de colas de destino

nombre_gestor_cola es el nombre del gestor de colas de destino

persistencia=persistencia_mensaje

Especifique una de las siguientes opciones:

- 0** La persistencia se define por la persistencia de colas predeterminada.
- 1** Los mensajes no son persistentes.
- 2** Los mensajes son persistentes.

Si la opción no se especifica o si se especifica de forma incorrecta, se utiliza la persistencia de colas predeterminada.

prioridad=prioridad_mensaje

Especifica la prioridad del mensaje. CICS soporta valores enteros en el rango 0 - 9 para las prioridades de mensajes, donde 9 se asigna a los mensajes de prioridad más alta y 0 se asigna a los mensajes de prioridad más baja. De lo contrario, especifique **-1** para utilizar la prioridad predeterminada que se define para la cola de destino.

destino_respuesta=cola_respuestas

Especifica la cola que se va a utilizar para el mensaje de respuesta. Si esta opción no se especifica, CICS utiliza una cola dinámica para el mensaje de respuesta. Debe definir la cola de respuesta en un objeto QLOCAL antes de utilizar esta opción.

servicio_destino_SOAPJMS=serie

Identifica el servicio de destino. Si CICS es el proveedor de servicios, el servicio de destino debe tener la forma `'/string'`, ya que CICS utiliza esto como la vía de acceso cuando intenta coincidir con URIMAP. Si no se especifica esta opción, se utiliza el valor que se especifica en TRIGDATA en la cola de entrada en el proveedor de servicios.

tiempo_espera=tiempo_espera

El tiempo de espera en milisegundos que espera el solicitante de servicio por

una respuesta. Si se especifica un valor de cero, o si se omite esta opción, la solicitud no sobrepasará el tiempo de espera.

timeToLive=hora_caducidad

Especifica la hora de caducidad para la solicitud en milisegundos. Si la opción no se especifica o si se especifica de forma incorrecta, la solicitud no caducará.

servicio_destino=serie

Identifica el servicio de destino. Si CICS es el proveedor de servicios, el servicio de destino debe tener la forma `'/string'`, ya que CICS utiliza esto como la vía de acceso cuando intenta coincidir con URIMAP. Si no se especifica esta opción, se utiliza el valor que se especifica en TRIGDATA en la cola de entrada en el proveedor de servicios.

Ejemplo

Este ejemplo muestra un URI para el transporte IBM MQ:

```
jms:/queue?destination=queue01@cics007&timeToLive=10&replyDestination=rqueue05&targetService=/myservice
```

Para obtener información sobre "connectionFactory" y "initialContextFactory", consulte la documentación de producto de *IBM MQ*.

Configuración de CICS para soportar mensajes permanentes

CICS proporciona soporte para enviar mensajes permanentes utilizando el protocolo de transporte de WebSphere MQ para una aplicación de proveedor de servicios web que se despliega en una región de CICS.

Acerca de esta tarea

CICS utiliza Business Transaction Services (BTS) para garantizar que se recuperen los mensajes permanentes en caso de un fallo del sistema CICS. Para que esto funcione correctamente, siga estos pasos:

Procedimiento

1. Utilice IDCAMS para definir el archivo de repositorio y la cola de solicitud local para MVS. Debe especificar un valor adecuado para STRINGS para la definición de archivo. El valor predeterminado de 1 no suele ser suficiente, se le recomienda que utilice 10.
2. Defina el archivo de repositorio y la cola de solicitud local para CICS. Los detalles de cómo definir la cola de solicitud local para CICS se describen en "Definición de colas locales en un proveedor de servicios" en la página 48. Debe especificar un valor adecuado para STRINGS en la definición de archivo. El valor predeterminado de 1 no suele ser suficiente, se le recomienda que utilice 10.
3. Defina un recurso PROCESSTYPE con el nombre DFHMQSOA, utilizando el nombre de archivo de repositorio como valor para la opción FILE.
4. Asegúrese de que durante el proceso de un mensaje permanente, un programa emite un mandato **EXEC CICS SYNCPOINT** antes de que se solicite el primer punto de sincronización implícito; por ejemplo, utilizando un mandato SPI como **EXEC CICS CREATE TDQUEUE** toma implícitamente un punto de sincronización. Emitiendo un mandato **EXEC CICS SYNCPOINT** confirma que el mensaje permanente se ha procesado correctamente. Si un programa no solicita explícitamente un punto de sincronización antes de intentar tomar implícitamente un punto de sincronización, se emite una terminación anormal ASP7.

Resultados

Qué hacer a continuación

Para mensajes de solicitud unidireccional, si el servicio web termina de forma anormal o vuelve atrás, se retiene información suficiente para permitir que un programa o transacción vuelva a intentar la solicitud que ha fallado o informar del fallo adecuadamente. Necesita proporcionar este programa o transacción de recuperación. Consulte “Proceso de mensajes permanentes” para obtener más detalles.

Proceso de mensajes permanentes

Cuando se recibe una solicitud de servicio web en un mensaje permanente de WebSphere MQ, CICS crea un proceso BTS exclusivo con el tipo de proceso DFHMQSOA. Los datos relacionados con la solicitud de entrada se capturan en contenedores de datos BTS asociados con el proceso.

El proceso se planifica para ejecutarse de forma asíncrona. Si el servicio web se completa correctamente y se envía, CICS suprime el proceso BTS. Esto incluye el caso en el que se genera un error SOAP y se devuelve al solicitante de servicio web.

Proceso de errores

Si se produce un error al crear el proceso BTS necesario, la transacción de servicio web termina anormalmente, y la solicitud de servicio web de entrada no se procesa. Si BTS no se puede utilizar, se emite el mensaje DFHPI0117 y CICS continúa sin BTS, utilizando el mecanismo de contenedor basado en el canal existente.

Si se produce un error de CICS antes de que el servicio web inicie o complete el proceso, la recuperación BTS garantiza que el proceso se vuelve a planificar cuando se reinicia CICS.

Si el servicio web finaliza anormalmente y vuelve atrás, el proceso BTS se marca como completo con un estado ABENDED. Para mensajes de solicitud que requieren una respuesta, se devuelve un error SOAP al solicitante de servicio web. El proceso BTS se cancela y CICS no retiene información sobre la solicitud fallida. CICS emite el mensaje DFHBA0104 en la cola de datos transitoria CSBA, y el mensaje DFHPI0117 en la cola de datos transitoria CPIO.

Para mensajes de dirección única, no hay forma de devolver información sobre el fallo al solicitante por lo que el proceso BTS se retiene en un estado COMPLETE ABENDED. CICS emite el mensaje DFHBA0104 en la cola de datos transitoria CSBA y el mensaje DFHPI0116 en la cola de datos transitoria CPIO.

Puede utilizar la transacción CBAM para visualizar procesos COMPLETE ABENDED, o puede proporcionar una transacción de recuperación para comprobar los procesos COMPLETE ABENDED de DFHMQSOA y llevar a cabo la acción adecuada.

Por ejemplo, la transacción de recuperación puede:

1. Restablecer el proceso BTS utilizando el mandato **RESET ACQPROCESS**.
2. Emitir el mandato **RUN ASYNC** para reintentar el servicio web que falla. Puede mantener un recuento de reentradas en otro contenedor de datos en el proceso para evitar un fallo repetido.

3. Utilizar información en los contenedores de datos asociados para informar del problema:

El contenedor de datos DFHMQORIGINALMSG contiene el mensaje recibido de WebSphere MQ, que puede contener cabeceras RFH2.

El contenedor de datos DFHMQMSG contiene el mensaje WebSphere MQ con las cabeceras RFH2 eliminadas.

El contenedor de datos DFHMQDLQ contiene el nombre de la cola de mensajes no entregados asociada al mensaje original.

El contenedor de datos DFHMQCONT contiene el bloque de control WebSphere MQ MQMD relacionado con **MQ GET** para el mensaje original.

Interoperabilidad entre el asistente de servicios web y WSRR

El asistente de servicios web CICS puede interoperar con IBM WebSphere Service Registry and Repository (WSRR). Utilice WSRR para encontrar los servicios web que está solicitando más rápidamente e imponer el control de versión de los servicios web que está proporcionando.

DFHLS2WS y DFHWS2LS incluyen parámetros para interoperar con WSRR. DFHLS2WS también incluye un parámetro opcional que permite añadir metadatos personalizados propios al documento WSDL en WSRR.

Si desea que el asistente de servicios web se comuniquen de forma segura con WSRR, puede utilizar el cifrado de nivel de sockets seguros (SSL). Tanto DFHLS2WS como DFHWS2LS incluyen parámetros para el uso de cifrado SSL.

Para utilizar SSL con el asistente de servicios web y WSRR, consulte “Ejemplo de cómo utilizar SSL con el asistente de servicios web y WSRR”.

Ejemplo de cómo utilizar SSL con el asistente de servicios web y WSRR

Puede interoperar de forma segura entre el asistente de servicios web y un servidor IBM WebSphere Service Registry and Repository (WSRR) utilizando el cifrado Capa de sockets seguros (SSL). Para utilizar el cifrado SSL, necesita un almacén de claves y un almacén de confianza; también debe especificar ciertos parámetros en el asistente de servicios web.

Acerca de esta tarea

Complete los pasos siguientes para utilizar el cifrado SSL para interacciones entre el asistente de servicios web y WSRR.

Procedimiento

1. Cree un almacén de claves para los certificados de claves privadas y claves públicas (PKC).
 - a. Puede crear un almacén de claves utilizando un programa de configuración de claves como IBM Key Management Utility (iKeyman).
 - b. Especifique el parámetro **SSL-KEYSTORE** en DFHWS2LS o DFHLS2WS con el nombre completo del almacén de claves que ha creado.
 - c. Opcional: Especifique el parámetro **SSL-KEYPWD** en DFHWS2LS o DFHLS2WS con la contraseña del almacén de claves que ha creado.
2. Cree un almacén de confianza para todos los certificados de entidad emisora de certificados raíz de confianza (CA). Estos certificados se utilizan para establecer la fiabilidad de los certificados de claves públicas de entrada.

- a. Puede crear un almacén de confianza utilizando un programa de configuración de claves como IBM Key Management Utility (iKeyman).
 - b. Especifique el parámetro **SSL-TRUSTSTORE** en DFHWS2LS o DFHLS2WS con el nombre completo del almacén de confianza que ha creado.
 - c. Opcional: Especifique el parámetro **SSL-TRUSTPWD** en DFHWS2LS o DFHLS2WS con la contraseña del almacén de confianza que ha creado.
3. Pruebe que el asistente de servicios web puede comunicarse con WSRR utilizando el cifrado SSL.
 - a. Puede utilizar los ejemplos de muestra proporcionados por IBM WebSphere Application Server para probar el asistente de servicios web con WSRR.
 - Los almacenes de claves de muestra proporcionados por WebSphere Application Server son DummyClientKeyFile.jks y DummyServerKeyFile.jks.
 - Los almacenes de confianza de ejemplos proporcionados por WebSphere Application Server son DummyClientTrustFile.jks y DummyServerTrustFile.jks.
 - b. Sustituya las claves en los archivos de almacén confianza y de claves de ejemplo. Estas claves se envían con WebSphere Application Server y deben sustituirse por seguridad.

Resultados

El asistente de servicios web puede utilizar ahora el cifrado SSL para comunicarse de forma segura con WSRR a través de una red.

Creación de la infraestructura de servicios web

Para desplegar un servicio web para CICS, debe crear la infraestructura de transporte necesaria y definir una o más interconexiones que procesarán las solicitudes de servicios web. Normalmente, una interconexión puede procesar solicitudes para muchos servicios web diferentes y, cuando despliega un nuevo servicio web en el sistema CICS, puede elegir utilizar una interconexión existente.

Infraestructura de servicios web

Las aplicaciones CICS en una región CICS pueden proporcionar un servicio a, o una petición de servicio desde, aplicaciones que son externas a esa región utilizando una interconexión de servicios web. Cuando CICS es un proveedor de servicios, la aplicación CICS proporciona un servicio a la aplicación externa. Cuando CICS es un solicitante de servicio, la aplicación externa proporciona un servicio a la aplicación CICS. Las interconexiones de servicios web se pueden configurar para utilizar zEnterprise Application Assist Processor (zAAP) donde esté disponible.

CICS como proveedor de servicios

Para que CICS proporcione un servicio a un solicitante de servicio externo, debe recibir la solicitud de servicio y pasarla a través de una interconexión al programa de aplicación de destino. La respuesta de la aplicación se devuelve al solicitante de servicio a través de la misma interconexión.

Figura 15 en la página 55 muestra una configuración de ejemplo de la arquitectura y los recursos necesarios para procesar una solicitud desde un solicitante de servicio externo cuando CICS es un proveedor de servicios que utiliza una interconexión Java.

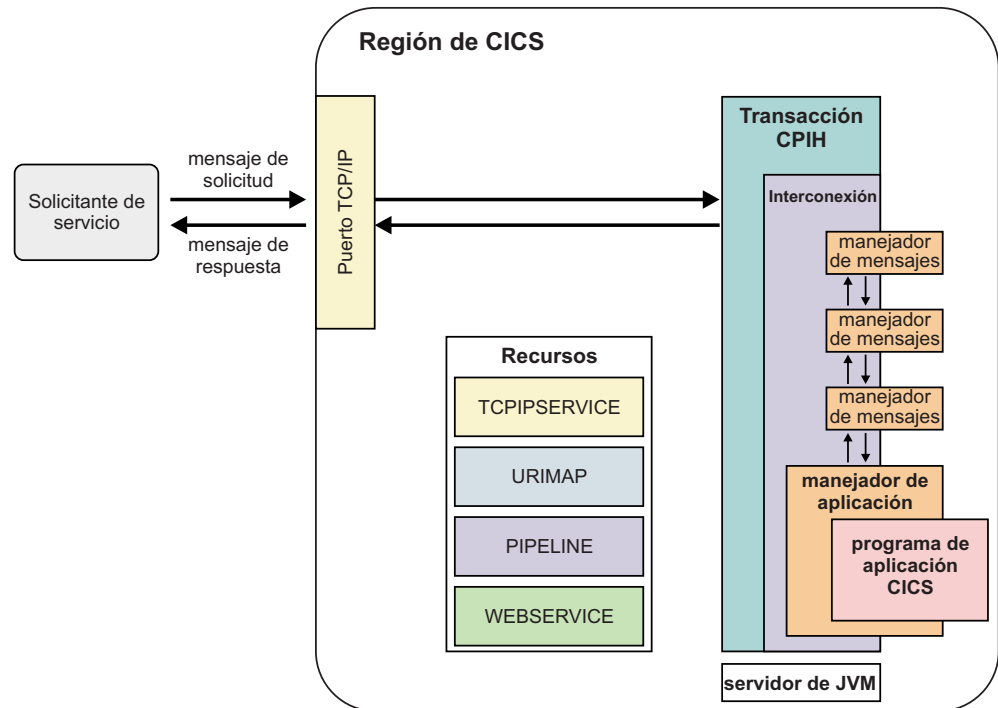


Figura 15. La arquitectura y recursos de un proveedor de servicios

Para procesar una solicitud, CICS debe realizar las siguientes operaciones:

1. Recibir la solicitud del solicitante de servicio.
El recurso TCPIPService especifica un puerto para las solicitudes entrantes. Este puerto es controlado por la transacción del proceso de escucha de sockets suministrado por CICS (CSOL).
2. Examinar la solicitud y extraer el contenido relevante para el programa de aplicación de destino.
Cuando el mensaje de solicitud se recibe en el puerto adecuado, las definiciones de recursos URIMAP se escanean para una definición URIMAP que tiene su atributo USAGE establecido en PIPELINE y su atributo PATH establecido en el URI encontrado en la solicitud. Si se encuentra una definición URIMAP adecuada, se utilizan las definiciones PIPELINE y WEBSERVICE de los atributos PIPELINE y WEBSERVICE de la definición URIMAP. El atributo TRANSACTION de la definición URIMAP determina el nombre de la transacción que debe anexarse al proceso de la interconexión. Se utiliza de forma predeterminada la transacción CPIH. La definición URIMAP también identifica los recursos PIPELINE y WEBSERVICE para utilizar. Estos recursos controlan el proceso que lleva a cabo CICS.
3. Invoque al programa de aplicación, pasando los datos extraídos de la solicitud.
Los manejadores de mensajes de la interconexión y el manejador de aplicación convierten el mensaje de solicitud a la estructura de mensaje de la aplicación que espera el programa de aplicación de proveedor de servicios. El programa procesa esta entrada y devuelve una respuesta al manejador de aplicación.
4. Construya una respuesta utilizando los datos devueltos por el programa de aplicación y envíe una respuesta al solicitante de servicio.

El manejador de aplicación y los manejadores de mensajes convierten el mensaje de respuesta recibido de la aplicación de proveedor de servicios a un mensaje en el formato de la solicitud original. Este mensaje se envía de nuevo al solicitante de servicio.

Se pueden realizar algunos procesos dentro de la interconexión utilizando zEnterprise Application Assist Processor (zAAP) si la interconexión está configurada adecuadamente. Para obtener más información, consulte el apartado “Interconexiones SOAP basadas en Java” en la página 57.

CICS como solicitante de servicio

Para que CICS invoque un servicio externo, un programa de aplicación envía una solicitud que pasa a través de una interconexión a un servicio de destino. La respuesta del servicio se devuelve al programa de aplicación a través de la misma interconexión.

Figura 16 muestra una configuración de ejemplo de la arquitectura y los recursos necesarios para procesar una solicitud desde un programa de aplicación CICS para datos de un proveedor de servicios externo a la región de CICS, utilizando una interconexión Java.

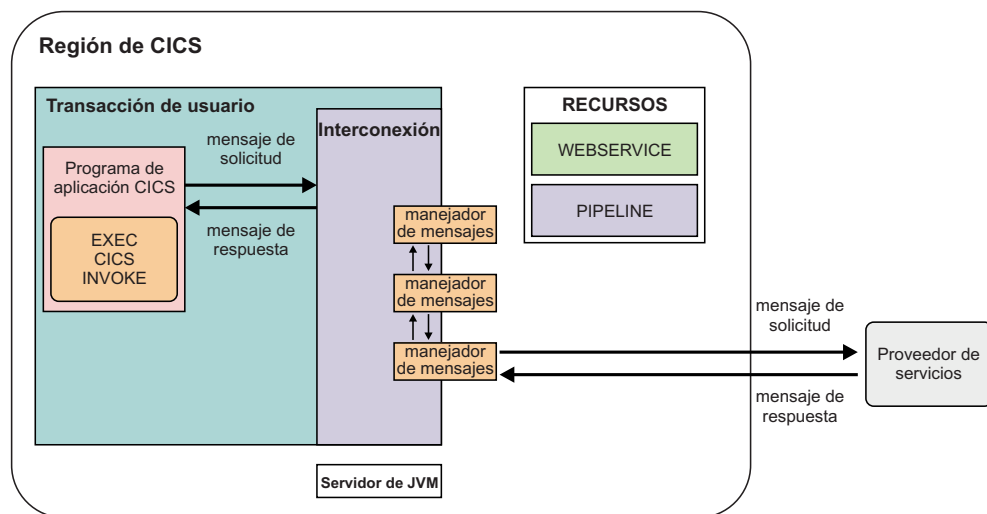


Figura 16. Arquitectura y recursos para un solicitante de servicio

Para procesar una solicitud, CICS debe realizar las siguientes operaciones:

1. Compilar una solicitud utilizando datos proporcionados por el programa de aplicación.

Cuando el programa de aplicación CICS inicia una solicitud para un proveedor de servicios externo a la región de CICS, la aplicación de solicitante invoca el mandato EXEC CICS INVOKE SERVICE. El mandato EXEC CICS INVOKE SERVICE invoca la interconexión. La interconexión convierte la estructura de lenguaje de aplicación en un lenguaje que puede procesar el proveedor de servicios, por ejemplo, un mensaje SOAP.

2. Enviar la solicitud al proveedor de servicios.

CICS envía el mensaje de solicitud al proveedor de servicios remoto utilizando HTTP o WebSphere MQ.

3. Recibir una respuesta del proveedor de servicios.

Cuando se recibe el mensaje de respuesta del proveedor de servicios, CICS pasa el mensaje de vuelta a la interconexión.

4. Examinar la respuesta y extraer el contenido relevante para el programa de aplicación original.

La interconexión convierte el mensaje de respuesta del proveedor de servicios en una estructura de lenguaje de aplicación, que se pasa al programa de aplicación. El control se devuelve al programa de aplicación.

Se pueden realizar algunos procesos dentro de la interconexión utilizando zEnterprise Application Assist Processor (zAAP) si la interconexión está configurada adecuadamente. Para obtener más información, consulte el apartado “Interconexiones SOAP basadas en Java”.

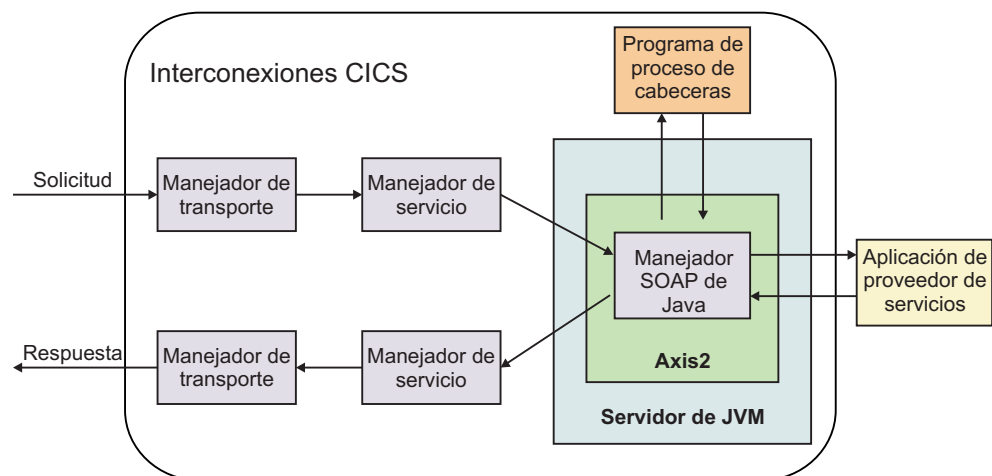
Interconexiones SOAP basadas en Java

CICS admite el uso de el motor SOAP basado en Java Axis2 para procesar las solicitudes de servicio web en la interconexión de solicitante y de proveedor. Como Axis2 utiliza Java, el procesamiento SOAP es elegible para descargarse al zEnterprise Application Assist Processor (zAAP).

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. Puede optar por utilizar Axis2 añadiendo un manejador SOAP de Java al archivo de configuración de interconexión y creando un servidor JVM para manejar el proceso de Axis2.

La habilitación de Axis2 no requiere volver a generar archivos de enlace para servicios web existentes que utilizan la interconexión. Los tiempos de respuesta pueden ser más lentos cuando se utiliza Axis2, pero puede descargar el proceso SOAP a zAAP. Para obtener más información sobre la descarga a zAAP, consulte Java support in CICS.

Cuando CICS es un proveedor de servicios, el manejador de terminal basado en Java utiliza Axis2 para analizar el sobre SOAP para un mensaje de solicitud. Puede utilizar programas de proceso de cabeceras para procesar cualquier cabecera SOAP asociada al mensajes SOAP. Axis2 también construye el mensaje de respuesta SOAP. Este proceso se muestra en el siguiente diagrama:



Cuando CICS es un solicitante de servicio, el manejador inicial basado en Java en la interconexión utiliza Axis2 para generar el sobre SOAP para un mensaje de solicitud. Puede utilizar programas de proceso de cabeceras para procesar cualquier cabecera SOAP asociada al mensajes SOAP. Axis2 también analiza el mensaje de respuesta SOAP.

Aplicaciones de servicio web y Java

Para interconexiones SOAP de modalidad de proveedor, los mensajes de solicitud y respuesta se pasan entre el controlador de terminal de la interconexión y la aplicación de servicio web utilizando un manejador de aplicación. El manejador de aplicación procesa el cuerpo de una solicitud SOAP para que la aplicación pueda utilizar la solicitud. El manejador de aplicación también genera una respuesta utilizando los datos devueltos de la aplicación. Si el controlador de terminal de la interconexión es un manejador de mensajes basado en Java, puede especificar el manejador de aplicación Axis2 proporcionado en el archivo de configuración de interconexión, en vez de especificar el manejador de aplicación DFHPITP proporcionado. El proceso de manejador de aplicación puede descargarse a zAAP. Para obtener más información sobre los manejadores de aplicación, consulte “Manejadores de aplicación” en la página 92.

Para interconexiones SOAP de modalidad de solicitante, la aplicación de servicio web invoca la interconexión utilizando el mandato **EXEC CICS INVOKE SERVICE**. Los mensajes de solicitud y respuesta se pasan entre la aplicación de servicio web y el manejador inicial en la interconexión. Si especifica un manejador basado en Java como el manejador inicial en la interconexión, entonces Axis2 procesa el mandato **EXEC CICS INVOKE SERVICE**, haciendo posible al descarga de este proceso a zAAP. Si el primer manejador no es un manejador basado en Java, el mandato **EXEC CICS INVOKE SERVICE** se procesa por CICS.

Proceso Axis2 en un servidor JVM

Axis2 requiere un servidor JVM, representado por un recurso JVMSERVER en CICS. El servidor JVM es un entorno de tiempo de ejecución que puede manejar varias solicitudes simultáneas desde diferentes programas Java en un único JVM. La vía de acceso de clase para el servidor JVM debe incluir los archivos de archivado Axis2 Java. Puede añadir automáticamente todos los archivos JAR necesarios a la vía de acceso de clase especificando la opción `JAVA_PIPELINE` en el perfil JVM. El archivo de configuración de interconexión también debe señalar al recurso JVMSERVER configurado para soportar Axis2.

Para obtener más información sobre servidores JVM, consulte Java support in CICS.

Manejador de cabeceras Axis2

Aunque puede utilizar programas de procesador de cabeceras existentes, es más eficaz escribir manejadores Axis2 en Java para procesar cabeceras SOAP. Estos manejadores también se pueden ejecutar en el servidor JVM y, por lo tanto, son elegibles para la descarga. Para obtener más información sobre la creación de manejadores Axis2, consulte Escribir su propio módulo Axis2.

Un programa de manejador de cabeceras puede utilizar las API de Axis2 para modificar o interactuar con el entorno Axis2, mensajes SOAP y servicios web individuales. No utilice estas API para personalizar Axis2, ya que puede cambiar Axis2 de manera que CICS no pueda ejecutar el motor correctamente. Los

manejadores Axis2 se soportan sólo si interactúan con el entorno Axis2 de forma que sea compatible con cómo CICS utiliza Axis2.

Repositorio de Axis2

Axis2 utiliza un repositorio para almacenar todos los archivos de configuración, servicios y módulos. CICS proporciona un repositorio predeterminado en el directorio *usshome/lib/pipeline/repository* en z/OS UNIX, donde *usshome* es el valor del parámetro de inicialización del sistema **USSHOME**.

El repositorio predeterminado contiene el archivo de configuración, *axis2.xml*, que necesita CICS para utilizar Axis2. Este archivo está en el subdirectorio */conf* en el repositorio. Si crea su propio repositorio, debe copiar este archivo en el repositorio para que CICS funcione con Axis2.

No edite el archivo *axis2.xml*, a menos que esté registrando programas de manejador. Este archivo está gestionado como una parte interna de CICS, por lo tanto, no debe hacer otros cambios a este archivo a menos que lo solicite el soporte de IBM.

Formateo de datos para servicios web

Diferentes tecnologías de CICS pueden generar datos JSON y XML que cumplen igualmente con la especificación pero que son físicamente diferentes. También pueden informar de errores que se encontraron en un mensaje de entrada en diferentes maneras, como resultado del orden en el que se aplican las comprobaciones para validar los datos.

CICS utiliza varias tecnologías diferentes para transformar automáticamente datos JSON y XML. Incluyen recursos z/OS Connect for CICS (variantes Java y no Java), Axis2 y PIPELINE. Estas tecnologías generan datos JSON y XML en un formato externo como dictan las especificaciones relevantes.

Puede haber varias formas de representar datos, que cumplen igualmente con la especificación pero que son físicamente diferentes. Las tecnologías de CICS siempre generan datos que son compatibles, pero puede que haya diferencias físicas entre ellos. Por ejemplo, si conmuta entre las opciones de Java y no Java z/OS Connect for CICS para JSON, es posible que detecte pequeñas diferencias en el JSON generado.

Tales diferencias pueden incluir diferentes mensajes de error que se están notificando bajo condiciones de error, diferencias en cómo se insertan los caracteres con espacios en blanco, representaciones alternativas (pero equivalentes) para datos numéricos y variaciones en cómo se permite el escape de caracteres especiales. Se pueden introducir más cambios de esta naturaleza como resultado de la aplicación de un mantenimiento correctivo para CICS, o con nuevas versiones de CICS.

Debe escribirse sistemas asociados, como un cliente JSON, para que soporten variaciones que cumplen con la especificación de esta naturaleza. Los socios a menudo aprovechan las bibliotecas y tecnologías interprofesionales de uso generalizado para interactuar con JSON y XML; como bibliotecas gestionadas automáticamente con diferencias de formato de poca importancia. Sin embargo, es posible para sistemas socio menos compatibles detectar y responder de forma diferente a las diferencias de formato en las distintas tecnologías CICS, por lo tanto, puede que sea necesario tener cuidado si está escribiendo aplicaciones que trabajan directamente con JSON o XML sin la ventaja de un analizador basado en estándares.

CICS como proveedor de servicios de solicitudes JSON

Para que CICS proporcione un servicio a un cliente JSON externo, debe recibir la solicitud de servicio y pasarla a través de una interconexión al programa de aplicación de destino. La respuesta de la aplicación se devuelve al cliente JSON a través de la misma interconexión.

Hay varias formas de configurar CICS como proveedor de servicios para solicitudes JSON:

- Usando z/OS Connect for CICS.

z/OS Connect for CICS es una tecnología para acceder a activos z/OS, como programas CICS, utilizando JSON. Para obtener más información sobre z/OS Connect for z/OS, y algunas diferencias entre interconexiones z/OS Connect for CICS y CICS Java, consulte *Getting started with z/OS Connect*.

- Utilizando interconexiones CICS Java.

Las interconexiones de CICS Java son la tecnología que se proporciona en versiones anteriores de CICS para permitir el acceso a programas CICS utilizando JSON. Para obtener más información sobre interconexiones de CICS Java, consulte *Java-based SOAP pipelines*.

- Utilizando las características JAX-RS y JSON del servidor JVM de CICS Liberty directamente.

- Utilizando el controlador de terminal proporcionado por CICS DFHPIJT.

Puede configurar una interconexión de proveedor con el controlador de terminal DFHPIJT. Esto permite a la interconexión procesar solicitudes JSON sin la necesidad de instalar un servidor JVM. Se aplican las siguientes restricciones:

Restricción:

- Los servicios web JSON de RESTful no se soportan.
- El conmutador de contexto en la interconexión no se soporta.
- No es posible utilizar los servicios web SOAP y JSON en una interconexión JSON. DFHPIJT sólo maneja mensajes JSON. La recepción de un mensaje SOAP da como resultado una respuesta de error.

CICS recibe datos JSON y los transforma en datos de aplicación estructurados que entiende el programa de aplicación CICS. Las respuestas de la aplicación CICS se transforman en una carga útil JSON para la respuesta de salida. Las transformaciones requieren el análisis de mensajes. Si utiliza z/OS Connect for CICS, tiene la opción de utilizar un analizador JSON basado en Java o un equivalente que no es Java. La opción de configuración se especifica en el archivo de configuración de interconexión de z/OS Connect for CICS. Para obtener más información sobre la configuración de z/OS Connect for CICS, consulte *Configuring z/OS Connect for CICS*. Si utiliza interconexiones CICS Java, el análisis sólo se lleva a cabo utilizando Java en el servidor JVM. Las implicaciones de las diferentes decisiones de configuración son:

- Si analiza JSON utilizando Java, el proceso es elegible para la descarga de un zEnterprise Application Assist Processor (zAAP) si está disponible. La descarga de un proceso puede tener un coste de beneficios.
- Si utiliza el analizador que no es Java z/OS Connect for CICS, algunas cargas de trabajo pueden conseguir ventajas de rendimiento. Para obtener más información sobre qué cargas de trabajo pueden beneficiarse, consulte el material de rendimiento que está disponible con este release. Aunque utilice el analizador que no es Java, la mayor parte del proceso de infraestructura de z/OS Connect for CICS es elegible para la descarga en zAAP.

- Si utiliza el controlador de terminal proporcionado por CICS DFHPIJT, algunas cargas de trabajo pueden conseguir ventajas de rendimiento. Cuando utiliza este método de procesamiento de solicitudes JSON, ninguna parte del proceso es elegible para la descarga en zAAP.

CICS como proveedor de servicios para solicitudes JSON que utilizan z/OS Connect for CICS:

Para que CICS proporcione un servicio para un cliente JSON externo, debe recibir la solicitud en z/OS Connect for CICS, transformar el mensaje JSON y pasarlo al programa de aplicación de destino. La respuesta de la aplicación se devuelve al cliente JSON a través del mismo mecanismo.

Figura 17 muestra una configuración de ejemplo de la arquitectura y los recursos que se necesitan para procesar una solicitud desde un cliente JSON externo cuando CICS es un proveedor de servicios que utiliza z/OS Connect for CICS.

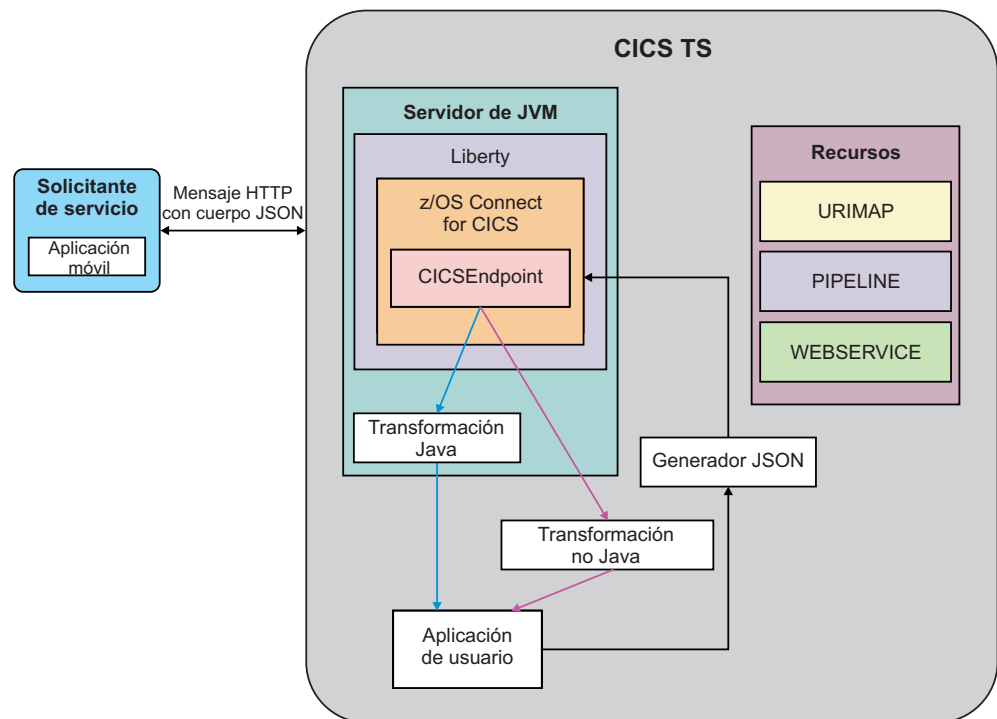


Figura 17. La arquitectura y recursos para un proveedor de servicios JSON que utiliza z/OS Connect for CICS

Proceso de una solicitud JSON con z/OS Connect for CICS

Para procesar una solicitud, CICS lleva a cabo las siguientes operaciones:

1. Recibir la solicitud del solicitante de servicio.
WebSphere Liberty recibe la solicitud y la pasa a z/OS Connect for CICS.
2. CICS resuelve el recurso URIMAP para la solicitud, explorando las definiciones URIMAP con un atributo USAGE establecido en JVMSERVER. URIMAP identifica el recurso WEBSERVICE que se está utilizando. Se inicia una nueva

tarea CICS para procesar la solicitud utilizando el ID de transacción desde el URIMAP. De forma predeterminada, se utiliza la transacción CPIH.

El recurso WEBSERVICE controla el proceso que lleva a cabo CICS. En concreto, se utiliza el archivo WSBInd señalado por el recurso WEBSERVICE para la transformación de datos entre los datos de aplicación estructurados y JSON. Los archivos WSBInd para servicios web JSON se generan utilizando los programas de utilidad DFHLS2JS y DFHJS2LS.

Nota: No se admite la validación en tiempo de ejecución de los datos JSON con el esquema. Se omite el valor del atributo VALIDATION de un recurso WEBSERVICE que se utiliza con una carga útil JSON.

Para obtener información sobre las restricciones que se aplican, consulte JSON web service restrictions.

3. z/OS Connect for CICS procesa la solicitud según cómo se ha configurado. Si z/OS Connect for CICS se configura para utilizar interceptores globales, los interceptores se ejecutan durante este proceso.
4. El CICS Endpoint recibe el control. La carga útil de JSON se transforma en datos de aplicación estructurados según la opción de configuración especificada en el archivo de configuración de interconexión. Hay dos opciones para llevar a cabo la transformación:
 - La transformación Java se realiza en el servidor JVM. Este proceso es elegible para descargar en un procesador zAAP si hay uno disponible.
 - La transformación no Java se realiza fuera del servidor JVM y puede proporcionar beneficios de rendimiento para determinadas cargas de trabajo. Para obtener más información sobre qué cargas de trabajo pueden beneficiarse, consulte el material de rendimiento que está disponible con este release.

La correlación se realiza según la información que hay en el archivo WSBInd. La salida de la transformación es equivalente en ambos casos.

5. CICS enlaza con el programa de aplicación y pasa los datos transformados. El programa procesa esta salida y devuelve una respuesta al generador JSON.
6. El generador JSON genera un mensaje de respuesta JSON utilizando la salida desde el paso 5. Este mensaje se envía de vuelta al solicitante de servicio a través de z/OS Connect for CICS.

z/OS Connect for CICS también proporciona una interfaz RESTful que soporta los métodos RESTful estándar:

POST
PUT
GET
DELETE
HEAD

Donde sea adecuado, esta interfaz utiliza la transformación y el generador para el que está configurado la interconexión, de la misma forma que hacen las solicitudes JSON normales. Para obtener más información sobre los servicios web de JSON RESTful, consulte Concepts of RESTful JSON web services.

La mayor parte del proceso de infraestructura de z/OS Connect for CICS es elegible para la descarga en un zEnterprise Application Assist Processor (zAAP).

CICS como proveedor de servicios para solicitudes JSON utilizando interconexiones CICS Java:

Para que CICS proporcione un servicio a un cliente JSON externo, debe recibir la solicitud de servicio y pasarla a través de una interconexión al programa de aplicación de destino. La respuesta de la aplicación se devuelve al cliente JSON a través de la misma interconexión. La transformación de JSON se lleva a cabo utilizando Java en el servidor JVM.

Figura 18 muestra una configuración de ejemplo de la arquitectura y los recursos necesarios para procesar una solicitud desde un cliente JSON externo cuando CICS es un proveedor de servicios que utiliza una interconexión Java. El proceso de interconexión para una solicitud JSON es similar a la forma en la que CICS procesa una solicitud SOAP en una interconexión Java. Para obtener más información, consulte el apartado Java-based SOAP pipelines.

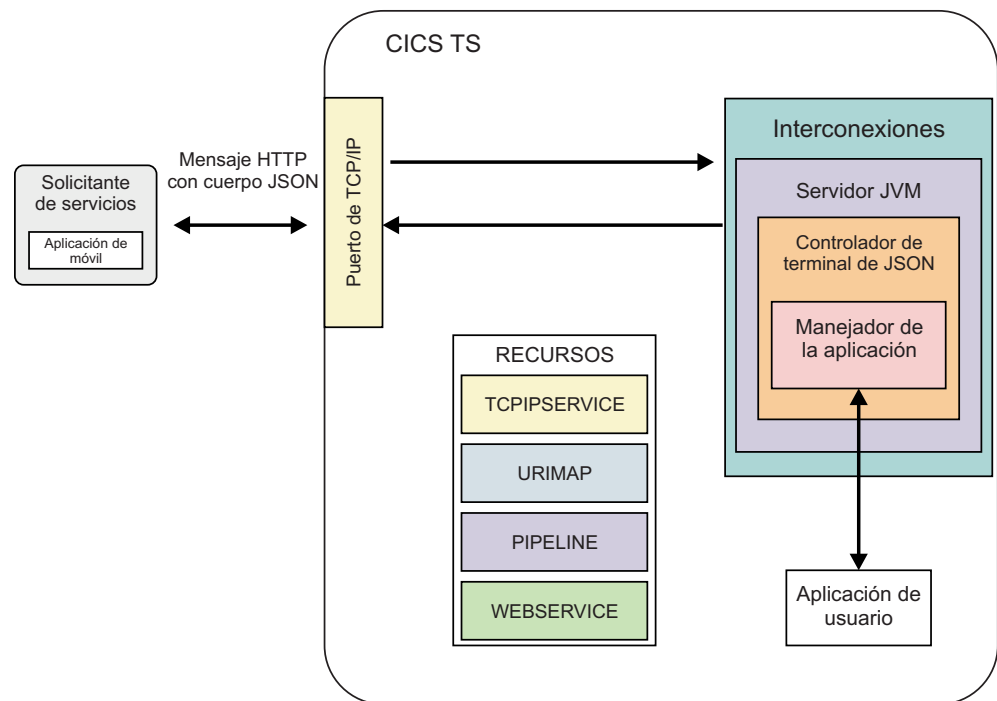


Figura 18. Arquitectura y recursos de un proveedor de servicios JSON que utiliza una interconexión CICS Java

Proceso de una solicitud JSON

Para procesar una solicitud, CICS lleva a cabo las siguientes operaciones:

1. Recibir la solicitud del solicitante de servicio.
El recurso TCPIP SERVICE especifica un puerto para las solicitudes entrantes. Este puerto está supervisado por la tarea del proceso de escucha de sockets proporcionado por CICS (CSOL).
2. Examinar la solicitud y extraer el contenido relevante para el programa de aplicación de destino.
Cuando el mensaje de solicitud se recibe en el puerto adecuado, las definiciones de recursos URIMAP se escanean para una definición URIMAP

que tiene su atributo USAGE establecido en PIPELINE y su atributo PATH establecido en el URI encontrado en la solicitud. Si se encuentra una definición URIMAP adecuada, se utilizan las definiciones PIPELINE y WEBSERVICE de los atributos PIPELINE y WEBSERVICE de la definición URIMAP. El atributo TRANSACTION de la definición URIMAP determina el nombre de la transacción que debe anexarse al proceso de la interconexión. Se utiliza de forma predeterminada la transacción CPIH. La definición URIMAP también identifica los recursos PIPELINE y WEBSERVICE para utilizar.

Estos recursos PIPELINE y WEBSERVICE controlan el proceso que lleva a cabo CICS. En concreto, el archivo WSBind al que señala el recurso WEBSERVICE se utiliza en la transformación de datos entre JSON y las estructuras de lenguaje. Los archivos WSBind para servicios web JSON se generan utilizando los programas de utilidad DFHLS2JS y DFHJS2LS.

Nota: No se admite la validación en tiempo de ejecución de los datos JSON con el esquema. Se omite el valor del atributo VALIDATION de un recurso WEBSERVICE que se utiliza con una carga útil JSON.

Para obtener información sobre las restricciones que se aplican, consulte JSON web service restrictions.

3. El proceso de interconexión comienza y la solicitud fluye a través de los manejadores definidos. No se espera que alguno de los manejadores que proporciona actualmente CICS para servicios web SOAP sea relevante para servicios web JSON.
4. Al final de la interconexión, se llama al controlador de terminal de JSON. Este controlador de terminal es un programa Java que se comunica con la interconexión Axis2. El controlador de terminal realiza la configuración necesaria de Axis2 y después inicia el motor de Axis2 con el cuerpo de solicitud HTTP. En la interconexión Axis2, se analiza el cuerpo JSON (si está presente) y se construye un modelo de objeto Java que representa el contenido. CICS llama al manejador de aplicación. La función principal del manejador de aplicación es correlacionar la representación del modelo de objeto Java de la solicitud con los datos de aplicación. Esta correlación se lleva a cabo utilizando la descripción de la estructura de lenguaje en el archivo WSBind.
5. Invoque al programa de aplicación, pasando datos que se extraen de la solicitud.

A continuación, el manejador de aplicación enlaza con el programa de aplicación. El programa procesa esta entrada y devuelve una respuesta al manejador de aplicación.

6. Construir una respuesta mediante los datos devueltos por el programa de aplicación y enviar una respuesta al solicitante de servicio.

El manejador de aplicación y los manejadores de mensajes convierten el mensaje de respuesta recibido de la aplicación de proveedor de servicios a un mensaje en el formato de la solicitud original. Este mensaje se envía de nuevo al solicitante de servicio.

Parte del proceso de la interconexión es elegible para su descarga en zEnterprise Application Assist Processor (zAAP).

Creación de la infraestructura de CICS para un proveedor de servicios SOAP

Para crear la infraestructura de CICS para un proveedor de servicios SOAP, tiene que crear un archivo de configuración de interconexión y varios recursos de CICS.

Antes de empezar

Si desea utilizar la interconexión Java, asegúrese de que existe un recurso JVMSERVER con la opción JAVA_PIPELINE=YES especificada en el perfil JVM.

Un servidor JVM puede manejar un proceso SOAP para muchas interconexiones Java.

Acerca de esta tarea

Puede definir el recurso PIPELINE en una región de CICS local utilizando las funciones CICS o CICSplex SM, o puede utilizar CICS Explorer para definir el recurso PIPELINE en una región de CICS local o en un paquete de CICS. Cuando utiliza CICS Explorer para definir un recurso PIPELINE en un paquete CICS, cree también el archivo de configuración de interconexión y empaquételo en el paquete CICS, así no tiene que gestionar este archivo por separado. Los recursos PROGRAM y WEBSERVICE también se pueden definir en paquetes CICS. Cuando define un recurso WEBSERVICE en un paquete CICS, puede importar un archivo de enlace de servicio web y un documento WSDL o archivo de archivado WSDL e inclúyalos en el paquete. También puede crear definiciones URIMAP para soportar el servicio web y empaquetarlas en un paquete. Para obtener más ayuda con el uso de CICS Explorer para crear y editar recursos en paquetes CICS, consulte Working with bundles in the CICS Explorer product documentation.

Procedimiento

1. Defina la infraestructura de transporte.
 - a. Si está utilizando el transporte WebSphere MQ, debe definir una o más colas locales que almacenan mensajes de entrada hasta que se procesan, y un proceso desencadenantes que especifica la transacción CICS que procesará los mensajes de entrada.
 - 1) Consulte Configuring CICS to use the WebSphere MQ transport para obtener más detalles.
 - b. Si está utilizando el transporte HTTP, debe definir un recurso TCPIPService que defina el puerto en el que se reciben las solicitudes de entrada.
 - 1) Consulte CICS resources for web services para obtener más detalles.
2. Opcional: Repita este paso para cada una de las configuraciones de transporte diferente que necesita.
3. Defina los manejadores de mensajes y los programas de proceso de cabeceras que desee incluir en el archivo de configuración de interconexión para procesar solicitudes de servicios web de entrada y sus respuestas. CICS proporciona los siguientes manejadores y programas de proceso de cabeceras:
 - a. SOAP message handlers, para procesar mensajes SOAP 1.1 o 1.2. Puede soportar sólo un nivel de SOAP en una interconexión de proveedor de servicio.
 - b. MTOM handler, para procesar mensajes MIME Multipart/Related que se ajustan a las especificaciones MTOM/XOP.
 - c. Support for securing web services, para procesar mensajes de servicio web seguro.
 - d. Support for Web Services Transactions, para procesar mensajes de transacción atómica.
4. Opcional: Si desea realizar su propio proceso en la interconexión, debe crear un manejador de mensajes o un programa de proceso de cabeceras. Consulte

Message handlers para obtener más detalles. Si decide crear programas de manejadores de mensajes personalizados, para optimizar el rendimiento deberá hacerlos de de proceso múltiple.

5. Cree un archivo de configuración de interconexión que contenga sus manejadores de mensajes, programas de proceso de cabeceras y manejador de aplicación.
 - a. CICS proporciona dos ejemplos de archivo de configuración de interconexión de modalidad de proveedor básicos, `basicsoap11provider.xml` y `basicsoap11javaprovider.xml`.
 - b. Puede editar estos ejemplo, o añadir manejadores de mensajes adicionales según proceda. Los ejemplos se proporcionan en la biblioteca `/usr/lpp/cicsts/cicsts54/samples/pipelines` (donde `/usr/lpp/cicsts/cicsts54` es el directorio de instalación predeterminado para los archivos de CICS en z/OS UNIX).
 - c. Para obtener más información sobre las opciones disponibles en el archivo de configuración de interconexión, consulte Pipeline configuration files
6. Copie el archivo de configuración de interconexión en un directorio adecuado en z/OS UNIX.
7. Cambie los permisos del configuración de interconexión para permitir que la región CICS lea el archivo.
8. Repita los pasos del 5 al 7 para cada configuración de interconexión diferente que necesite.
9. Cree un recurso PIPELINE.
 - a. El recurso PIPELINE define la ubicación del archivo de configuración de interconexión. También especifica un *directorio de recogida*, que es el directorio z/OS UNIX que contiene los archivos de enlace de servicio web y, opcionalmente, el WSDL.
 - b. Repita este paso para cada una de las diferentes configuraciones de interconexión.
 - a. Cuando cree un recurso PIPELINE, CICS lee cualquier archivo en el directorio de recogida especificado y crear el recurso WEBSERVICE y el recurso URIMAP dinámicamente.
10. A menos que utilice las definiciones PROGRAM instaladas automáticamente, cree un recurso PROGRAM para cada programa que se ejecute en la interconexión. Esto incluye los programas de aplicación de destino, que normalmente se ejecutan bajo la transacción CPIH. La transacción se define con el atributo TASKDATALOC(ANY). Por lo tanto, cuando enlace-edite el programa, deberá especificar la opción AMODE(31).

Resultados

El sistema CICS contiene ahora la infraestructura necesaria para cada proveedor de servicios.

Qué hacer a continuación

Podrá ampliar la configuración cuando lo necesite, bien para definir infraestructura de transporte adicional o para crear interconexiones adicionales.

Creación de la infraestructura CICS para un solicitante de servicio SOAP

Para crear la infraestructura de CICS para un solicitante de servicio SOAP, debe crear un archivo de configuración de interconexión y crear un número de recursos de CICS.

Antes de empezar

Si desea utilizar una interconexión Java, asegúrese de que existe un recurso JVMSERVER con la opción JAVA_PIPELINE=YES especificada en el perfil JVM. Consulte el apartado JVMSERVER resources.

Un servidor JVM puede manejar un proceso SOAP para muchas interconexiones Java.

Acerca de esta tarea

Puede definir el recurso PIPELINE en una región de CICS local utilizando las funciones CICS o CICSplex SM, o puede utilizar CICS Explorer para definir el recurso PIPELINE en una región de CICS local o en un paquete de CICS. Cuando utiliza CICS Explorer para definir un recurso PIPELINE en un paquete CICS, cree también el archivo de configuración de interconexión y empaquételo en el paquete CICS, así no tiene que gestionar este archivo por separado. Los recursos PROGRAM, WEBSERVICE y URIMAP también se pueden definir en paquetes CICS. Cuando define un recurso WEBSERVICE en un paquete CICS, puede importar un archivo de enlace de servicio web y un documento WSDL o archivo de archivado WSDL y empaquetarlos en un paquete, y para un proveedor de servicios puede elegir incluir una definición PROGRAM. También puede crear definiciones URIMAP para soportar el servicio web y empaquetarlas en un paquete. Para obtener más ayuda con el uso de CICS Explorer para crear y editar recursos en paquetes CICS, consulte Working with bundles in the CICS Explorer product documentation.

Procedimiento

1. Defina los manejadores de mensajes y los programas de proceso de cabeceras que desee incluir en el archivo de configuración de interconexión para procesar solicitudes de servicios web de entrada y sus respuestas. CICS proporciona los siguientes manejadores y programas de proceso de cabeceras:
 - a. Manejadores de mensajes SOAP, para procesar mensajes SOAP 1.1 o 1.2. Sólo puede soportar un nivel de SOAP en una interconexión de solicitante de servicio.
 - b. Manejador MTOM, para procesar mensajes MIME Multipart/Related que se ajuntan a las especificaciones MTOM/XOP.
 - c. Manejador de seguridad, para procesar mensajes de servicio web seguro.
 - d. Programa de proceso de cabeceras WS-AT, para procesar mensajes de transacción atómica.
2. Opcional: Si desea realizar su propio proceso en la interconexión, debe crear un manejador de mensajes o un programa de proceso de cabeceras. Consulte "Manejadores de mensajes" en la página 136 para obtener más detalles. Si decide crear programas de manejadores de mensajes personalizados, para optimizar el rendimiento deberá hacerlos de de proceso múltiple.
3. Cree un archivo de configuración de interconexión XML que contenga sus manejadores de mensajes y programas de proceso de cabeceras. CICS

proporciona dos ejemplos de archivos de configuración de interconexión de modalidad de solicitante básicos, `basicsoap11provider.xml` y `basicsoap11javaprovider.xml`, que puede copiar y editar según corresponda. Estos ejemplos se proporcionan en la biblioteca `/usr/lpp/cicsts/cicsts54/samples/pipelines` (donde `/usr/lpp/cicsts/cicsts54` es el directorio de instalación predeterminado para archivos CICS en z/OS UNIX). Para obtener más información sobre las opciones disponibles en el archivo de configuración de interconexión, consulte “Archivos de configuración de interconexión” en la página 81

4. Copie el archivo de configuración de interconexión en un directorio adecuado en z/OS UNIX.
5. Cambie los permisos de archivo de configuración de interconexión para permitir que la región de CICS lea el archivo.
6. Repita los pasos del 3 al 5 para cada configuración de interconexión diferente que necesite.
7. Cree un recurso PIPELINE. Consulte el apartado PIPELINE resources. El recurso PIPELINE define la ubicación del archivo de configuración de interconexión. También especifica un *directorio de recogida*, que es el directorio z/OS UNIX que contiene los archivos de enlace de servicio web y, opcionalmente, el WSDL. También puede especificar un tiempo de espera en segundos, que determina cuánto tiempo espera CICS una respuesta de los proveedores de servicios web. Repita este paso para cada archivo de configuración de interconexión. Cuando crea un recurso PIPELINE, CICS lee cualquier archivo en el directorio de recogida especificado y crea los recursos WEBSERVICE de forma dinámica (consulte WEBSERVICE resources).
8. A menos que utilice las definiciones PROGRAM instaladas automáticamente, cree un recurso PROGRAM para cada programa que se ejecuta en la interconexión. Consulte el apartado PROGRAM resources. Estos programas incluyen el programa de aplicación de solicitante de servicio, que normalmente se ejecuta bajo la transacción CPIH. La transacción se define con el atributo `TASKDATALOC(ANY)`. Por lo tanto, cuando enlaza-edita el programa, debe especificar la opción `AMODE(31)`.
9. Opcional: Cree un recurso URIMAP (consulte URIMAP resources) para obtener las solicitudes de cliente para cada URI que utilicen sus solicitantes de servicio para hacer solicitudes, siguiendo las instrucciones en Creating a URIMAP resource for CICS as a HTTP client. Puede especificar el URI directamente en el mandato **INVOKE SERVICE** en sus programas, en lugar de utilizar un recurso URIMAP. Sin embargo, utilizar un recurso URIMAP significa que no necesita recompilar las aplicaciones si el URI de un proveedor de servicios cambia. Con un recurso URIMAP, también puede elegir implementar la técnica de agrupación de conexiones, donde CICS mantiene la conexión de cliente abierta después de utilizarla, así la aplicación puede volver a utilizarla para solicitudes posteriores, u otra aplicación puede volver a utilizarla para llamar al mismo servicio.

Resultados

El sistema CICS contiene ahora la infraestructura necesaria para cada solicitante de servicio.

Qué hacer a continuación

Puede ampliar la configuración cuando necesite hacerlo, para crear interconexiones adicionales.

Creación de la infraestructura de CICS para un proveedor de servicios JSON

Para crear la infraestructura de CICS para un proveedor de servicios JSON, tiene que crear un archivo de configuración de interconexión y varios recursos de CICS.

Antes de empezar

Para utilizar CICS como un proveedor de servicios para solicitudes JSON o para utilizar la interfaz enlazable para transformar JSON, defina e instale un recurso JVMSERVER con un perfil JVM que tiene la opción **JAVA_PIPELINE=YES** especificada. Se proporciona una definición de recurso JVMSERVER de ejemplo denominada DFHAXIS en el grupo DFH\$AXIS.

Nota: La infraestructura descrita aquí supone que no está utilizando z/OS Connect for CICS para conectarse a su proveedor de servicios JSON y, por lo tanto, utiliza el análisis de Java en el servidor JVM para analizar mensajes JSON. Si desea utilizar un análisis JSON que no sea Java, debe utilizar z/OS Connect for CICS para conectarse al servicio web JSON. Para obtener información sobre la configuración de z/OS Connect for CICS, consulte *Configuring z/OS Connect for CICS*.

Procedimiento

1. Defina la infraestructura de transporte.
Defina un recurso TCPIP SERVICE que defina el puerto en el que se reciben las solicitudes de entrada. Consulte Recursos CICS para servicios web para obtener detalles.
2. Defina los manejadores de mensajes que desee incluir en el archivo de configuración de interconexión para procesar solicitudes de servicios web de entrada y sus respuestas.
Si desea realizar su propio proceso en la interconexión, deberá crear un manejador de mensajes. Consulte Manejadores de mensajes para obtener más detalles. Si decide crear programas de manejadores de mensajes personalizados, para optimizar el rendimiento deberá hacerlos de proceso múltiple.
3. Cree un archivo de configuración de interconexión que contenga sus manejadores de mensajes, programas de proceso de cabeceras y manejador de aplicación. CICS proporciona un archivo de configuración de interconexión en modalidad de proveedor de ejemplo, `jsonjavaprovider.xml`. Puede editar esta muestra para añadir manejadores de mensajes adicionales según proceda. Este ejemplo se proporciona en el directorio `/usr/lpp/cicsts/cicsts54/samples/pipelines`, donde `/usr/lpp/cicsts/cicsts54` es el directorio de instalación predeterminado para archivos CICS en z/OS UNIX). Para obtener más información acerca de las opciones disponibles en el archivo de configuración de interconexión, consulte “Elementos utilizados en las interconexiones de solicitante de servicio y proveedor de servicios” en la página 99.
4. Copie el archivo de configuración de interconexión en un directorio adecuado en z/OS UNIX.
5. Cambie los permisos del configuración de interconexión para permitir que la región CICS lea el archivo.
6. Cree un recurso PIPELINE. El recurso PIPELINE define la ubicación del archivo de configuración de interconexión. Especifica también un *directorio de recogida*, que es el directorio z/OS UNIX que contiene los archivos de enlace de servicio web. Repita este paso para cada una de las diferentes configuraciones de interconexión. Cuando instale un recurso PIPELINE o realice un PIPELINE

SCAN, CICS lee los archivos .wsbind en el directorio de recogida especificado y crea recursos WEBSERVICE y URIMAP adecuados de forma dinámica.

7. A menos que utilice las definiciones PROGRAM instaladas automáticamente, cree un recurso PROGRAM para cada programa que se ejecute en la interconexión. Esto incluye los programas de aplicación de destino, que normalmente se ejecutan bajo la transacción CPIH. La transacción se define con el atributo TASKDATALOC(ANY). Por lo tanto, cuando enlace-edite el programa, deberá especificar la opción AMODE(31).

Resultados

Ha creado la infraestructura necesaria para cada proveedor de servicios y ahora podrá instalar estos recursos en su sistema CICS.

Qué hacer a continuación

Instale los recursos. Podrá ampliar la configuración cuando lo necesite, bien para definir infraestructura de transporte adicional o para crear interconexiones adicionales.

Creación de la infraestructura de CICS para un proveedor de servicios JSON que no es de Java

Puede configurar un entorno que no es de Java para procesar solicitudes JSON. Para crear la infraestructura de CICS para un proveedor de servicios JSON que no es de Java, debe crear un archivo de configuración de interconexión y varios recursos CICS.

Procedimiento

1. Defina la infraestructura de transporte.
Defina un recurso TCPIPService que defina el puerto en el que se reciben las solicitudes de entrada. Consulte Recursos CICS para servicios web para obtener detalles.
2. Defina los manejadores de mensajes que desee incluir en el archivo de configuración de interconexión para procesar solicitudes de servicios web de entrada y sus respuestas.
Si desea realizar su propio proceso en la interconexión, deberá crear un manejador de mensajes. Consulte Manejadores de mensajes para obtener más detalles. Si decide crear programas de manejadores de mensajes personalizados, para optimizar el rendimiento deberá hacerlos de proceso múltiple.
3. Cree un archivo de configuración de interconexión XML que contiene los manejadores de mensajes.
En el archivo de configuración, debe especificar el programa del controlador de terminal DFHPIJT en un elemento <terminal_handler> como se muestra en Figura 19 en la página 71. DFHPIJT es el programa de manejador JSON proporcionado por CICS que permite el proceso de mensaje JSON que no son de Java.

```

| <service>
|   <terminal_handler>
|     <handler>
|       <program>DFHPIJT</program><handler_parameter_list/>
|     </handler>
|   </terminal_handler>
| </service>

```

Figura 19. Especificación del controlador de terminal DFHPIJT para un proceso no Java de mensajes JSON

Nota: Cuando utiliza DFHPIJT como el controlador de terminal, no define un manejador de aplicación en el archivo de configuración de interconexión, es decir, el archivo de configuración de interconexión no debe contener un elemento <apphandler>. Si se especifica un manejador de aplicación, no se invoca.

Para obtener más información acerca de las opciones disponibles en el archivo de configuración de interconexión, consulte “Elementos utilizados en las interconexiones de solicitante de servicio y proveedor de servicios” en la página 99.

4. Copie el archivo de configuración de interconexión en un directorio adecuado en z/OS UNIX.
5. Cambie los permisos del configuración de interconexión para permitir que la región CICS lea el archivo.
6. Cree un recurso PIPELINE. El recurso PIPELINE define la ubicación del archivo de configuración de interconexión. Especifica también un *directorio de recogida*, que es el directorio z/OS UNIX que contiene los archivos de enlace de servicio web. Repita este paso para cada una de las diferentes configuraciones de interconexión. Cuando instale un recurso PIPELINE o realice un PIPELINE SCAN, CICS lee los archivos .wsbind en el directorio de recogida especificado y crea recursos WEBSERVICE y URIMAP adecuados de forma dinámica.
7. A menos que utilice las definiciones PROGRAM instaladas automáticamente, cree un recurso PROGRAM para cada programa que se ejecute en la interconexión. Esto incluye los programas de aplicación de destino, que normalmente se ejecutan bajo la transacción CPIH. La transacción se define con el atributo TASKDATALOC(ANY). Por lo tanto, cuando enlace-edite el programa, deberá especificar la opción AMODE(31).

Resultados

Ha creado la infraestructura necesaria para cada proveedor de servicios y ahora podrá instalar estos recursos en su sistema CICS.

Qué hacer a continuación

Instale los recursos. Podrá ampliar la configuración cuando lo necesite, bien para definir infraestructura de transporte adicional o para crear interconexiones adicionales.

Configuración de z/OS Connect for CICS

Configuración de z/OS Connect for CICS 1.0

z/OS Connect for CICS 1.0 se distribuye como parte de CICS Transaction Server. Debe configurar un servidor JVM y establecer la configuración de la interconexión y los recursos para z/OS Connect, antes puede desplegar los servicios JSON. La configuración inicial es una actividad de una sola vez.

Antes de empezar

¿Ya tiene un servidor WebSphere Liberty JVM configurado en CICS? Aunque es posible alojar z/OS Connect y otros servicios no relacionados en el mismo entorno WebSphere Liberty, se recomienda configurar un servidor JVM individual para uso exclusivo de z/OS Connect. También se recomienda tener sólo un único servidor WebSphere Liberty JVM configurado en una única región de CICS.

Puede alojar z/OS Connect for CICS 1.0 en su propia región de CICS o un grupo de regiones de CICS, y utilizar el mecanismo de Enlace de programa distribuido para invocar programas CICS en las regiones de CICS que pertenecen a la aplicación.

Procedimiento

1. Cree un JVMSERVER y configúrelo para dar soporte a WebSphere Liberty. Para obtener más información sobre la creación de un JVMSERVER de WebSphere Liberty, consulte .
2. Configuración de WebSphere Liberty para los requisitos de seguridad. De forma predeterminada, WebSphere Liberty espera el uso de certificados SSL certificados por el cliente. Para habilitar la autenticación básica HTTP, añada la siguiente opción de configuración al archivo `server.xml`:

```
<!-- Allow fail-over to HTTP Basic Authentication -->
<webAppSecurity allowFailOverToBasicAuth="true"/>
```

Debe otorgar a los usuarios de z/OS Connect, el rol de seguridad **zosConnectAccess**. Para obtener más información sobre la seguridad de WebSphere Liberty, consulte *Configuring security for a Liberty JVM server*, y para la seguridad de z/OS Connect, consulte *Security for z/OS Connect*.

3. Actualice la lista `<featureManager>` en el archivo `server.xml` para el entorno WebSphere Liberty para incluir la característica `<feature>cicsts:zosConnect-1.0</feature>`, como muestra el siguiente ejemplo:

```
<featureManager>
  <feature>cicsts:core-1.0</feature>
  <feature>ssl-1.0</feature>
  <feature>cicsts:zosConnect-1.0</feature>
</featureManager>
```

4. Defina el controlador de servicio z/OS Connect for CICS 1.0 añadiendo la siguiente sentencia al archivo `server.xml`:

```
<com.ibm.cics.wlp.zosconnect.CICSEndpoint
  id="com.ibm.cics.wlp.zosconnect.CICSEndpointService"/>
```

5. Instale el JVMSERVER. Compruebe el archivo `messages.log` generado para obtener los mensajes de error o de aviso. Este registro contiene los mensajes generados por el servidor WebSphere Liberty, incluidos los mensajes devueltos por z/OS Connect for CICS 1.0, como los siguientes:

```
SRVE0169I: Loading Web Module: z/OS Connect.
SRVE0250I: Web Module z/OS Connect has been bound to default_host.
```

6. Cree un archivo de configuración de interconexión XML. El archivo de configuración de interconexión de ejemplo `jsonzosconnectprovider.xml` se proporciona en el directorio `/usr/lpp/cicsts/cicsts54/samples/pipelines/` (donde `/usr/lpp/cicsts/cicsts54` es el directorio de instalación predeterminado para los archivos CICS en z/OS UNIX). Debe decidir si desea analizar el JSON utilizando Java en el servidor Liberty JVM (que es el valor predeterminado), o utilizar el analizador JSON que no es Java:

- Para analizar el JSON utilizando Java en el servidor Liberty JVM, puede utilizar el archivo de configuración de interconexión de ejemplo, pero sustituya DFHWLP en el elemento <jvmserver> por el nombre de JVMSERVER desde el Paso 1.
- Para analizar el JSON utilizando el analizador que no es Java, modifique el archivo de configuración de ejemplo para añadir el atributo `java_parser="no"` al elemento <provider_pipeline_json> como en el siguiente ejemplo:

```
<?xml version="1.0" encoding="EBCDIC-CP-US"?>
<provider_pipeline_json java_parser="no"
  xmlns="http://www.ibm.com/software/http/cics/pipeline">
  <jvmserver>DFHWLP</jvmserver>
</provider_pipeline_json>
```

Sustituya DFHWLP por el nombre de JVMSERVER que ha creado al inicio de este procedimiento.

7. Copie el archivo de configuración en un directorio adecuado en zFS y asegúrese de que los permisos de archivo permiten a la región de CICS leer el archivo. Para obtener información, consulte Archivos de configuración de interconexión.
8. Cree un recurso PIPELINE. El recurso PIPELINE define la ubicación del archivo de configuración de interconexión en el atributo CONFIGFILE.
9. Opcional: Cree un recurso URIMAP para z/OS Connect. Los recursos URIMAP se utilizan para asociar una TRANSACTION y el ID de usuario predeterminado con trabajo de z/OS Connect. Se pueden utilizar uno o más recursos URIMAP para configurar una política predeterminada para z/OS Connect. Para obtener un ejemplo de configuración URIMAP y más información sobre las opciones de configuración, consulte :

Nota:

z/OS Connect realiza una autenticación adicional para solicitudes HTTP individuales, así las tareas de aplicación que se ejecutan en CICS están normalmente asociadas a un ID de usuario más específico que el ID de usuario inicial de URIMAP. Este ID de usuario inicial está sólo en vigor hasta que tiene lugar la autenticación específica de usuario en z/OS Connect.

Resultados

Su instancia de z/OS Connect for CICS 1.0 está configurada. Puede probar la configuración básica escribiendo esta URL en un navegador web: `https://hostname:portnumber/zosConnect/apim/services`, donde *hostname* es la dirección IP o nombre de host del sistema en el que se está ejecutando la región de CICS que está alojando z/OS Connect for CICS 1.0, y *portnumber* es el **httpsPort** que se especifica en el elemento <httpEndpoint> del archivo `server.xml`. El navegador web muestra una lista de los servicios instalados; como aún no hay servicios instalados, la lista está vacía.

Si recibe una respuesta HTTP 403 "AuthorizationFailed" en lugar de la lista de servicios esperada, revise la configuración de seguridad del Paso 2. Es probable que el usuario autenticado no esté autorizado para utilizar z/OS Connect.

Qué hacer a continuación

Está preparado para desplegar servicios web JSON en z/OS Connect for CICS 1.0.

Configuración de z/OS Connect Enterprise Edition

z/OS Connect Enterprise Edition es un producto que se debe solicitar de forma individual; no se proporciona como parte de CICS TS. En primer lugar, debe instalar el componente de tiempo de ejecución de z/OS Connect Enterprise Edition y configurar un archivo zFS para que CICS pueda localizarlo. Después, configure un servidor JVM y establezca la configuración de la interconexión y los recursos para z/OS Connect, antes puede desplegar los servicios de JSON y las API. Esta configuración inicial es una actividad única que permite a z/OS Connect Enterprise Edition ejecutarse incorporado en el servidor Liberty que se distribuye con CICS TS.

Antes de empezar

Instale el tipo de ejecución de z/OS Connect Enterprise Edition. Siga las instrucciones en *z/OS Connect Enterprise Edition documentation* in IBM Knowledge Center, pero tenga en cuenta que la adaptación de z/OS Connect Enterprise Edition para ejecutarse incorporado en CICS significa que no necesita crear una instancia de servidor Liberty individual. Sólo lo necesita instalado en la medida que los componentes del sistema de archivos están disponibles en zFS. La instalación del Editor de API proporcionado con z/OS Connect Enterprise Edition no es parte de esta tarea.

¿Ha instalado anteriormente z/OS Connect for CICS 1.0? Si es así, varios de los pasos de esta tarea no son necesarios. Los pasos que puede omitir se indican en la lista.

Procedimiento

1. Añada ZCEE_INSTALL_DIR a las opciones del servidor JVM. Para obtener más información sobre la opción y un ejemplo, consulte *Opciones del servidor JVM*.
2. Cree un JVMSERVER y configúrelo para soportar WebSphere Liberty. Para obtener más información sobre la creación de un JVMSERVER de WebSphere Liberty, consulte .

Si ha instalado anteriormente z/OS Connect for CICS 1.0, puede saltarse este paso porque es probable que ya tenga una configuración adecuada.

3. Configuración de WebSphere Liberty para los requisitos de seguridad. De forma predeterminada, espera el uso de certificados SSL certificados por el cliente. Para habilitar la autenticación básica HTTP, añada la siguiente opción de configuración al archivo `server.xml`:

```
<!-- Allow fail-over to HTTP Basic Authentication -->
<webAppSecurity allowFailOverToBasicAuth="true"/>
```

Debe otorgar a los usuarios de z/OS Connect el rol de seguridad `zosConnectAccess` . Para obtener más información sobre la seguridad de WebSphere Liberty, consulte *Configuring security for a Liberty JVM server*, y para la seguridad de z/OS Connect, consulte *Security for z/OS Connect*.

Si ha instalado anteriormente z/OS Connect for CICS 1.0, puede saltarse este paso porque es probable que ya tenga una configuración adecuada.

4. Actualice la lista `<featureManager>` en el archivo `server.xml` para WebSphere Liberty para que incluya una característica `<feature>cicsts:zosConnect-2.0</feature>`, como se muestra en el siguiente ejemplo:

```
<featureManager>
  <feature>cicsts:core-1.0</feature>
  <feature>ssl-1.0</feature>
  <feature>cicsts:zosConnect-2.0</feature>
</featureManager>
```

Nota: Las características z/OS Connect for CICS 1.0 y z/OS Connect Enterprise Edition se excluyen mutuamente. Si está migrando desde z/OS Connect for CICS 1.0 a z/OS Connect Enterprise Edition, cambie la característica `cicsts:zosConnect-1.0` existente a `cicsts:zosConnect-2.0`.

5. Defina el controlador de servicio z/OS Connect añadiendo la siguiente sentencia al archivo `server.xml`:

```
<com.ibm.cics.wlp.zosconnect.CICSEndpoint
  id="com.ibm.cics.wlp.zosconnect.CICSEndpointService"/>
```

Si ha instalado anteriormente z/OS Connect for CICS 1.0, puede saltarse este paso porque es probable que ya tenga una configuración adecuada.

6. Instale el JVMSERVER. Compruebe el archivo `messages.log` generado para obtener los mensajes de error o de aviso. Este registro contiene los mensajes generados por el servidor WebSphere Liberty, incluidos los mensajes devueltos por z/OS Connect Enterprise Edition como el siguiente:

```
SRVE0169I: Loading Web Module: z/OS Connect.
```

```
SRVE0250I: Web Module z/OS Connect has been bound to default_host.
```

7. Cree un archivo de configuración de interconexión XML. El archivo de configuración de interconexión de ejemplo `jsonzosconnectprovider.xml` se proporciona en el directorio `/usr/lpp/cicsts/cicsts54/samples/pipelines/` (donde `/usr/lpp/cicsts/cicsts54` es el directorio de instalación predeterminado para los archivos CICS en z/OS UNIX). Debe decidir si desea analizar el JSON utilizando Java en el servidor Liberty JVM (que es el valor predeterminado), o utilizar el analizador JSON que no es Java:

- Para analizar el JSON utilizando Java en el servidor Liberty JVM, puede utilizar el archivo de configuración de interconexión de ejemplo, pero sustituya DFHWLP en el elemento `<jvmserver>` por el nombre de JVMSERVER desde el Paso 2.
- Para analizar el JSON utilizando el analizador que no es Java, modifique el archivo de configuración de ejemplo para añadir el atributo `java_parser="no"` al elemento `<provider_pipeline_json>` como en el siguiente ejemplo:

```
<?xml version="1.0" encoding="EBCDIC-CP-US"?>
<provider_pipeline_json java_parser="no"
  xmlns="http://www.ibm.com/software/http/cics/pipeline">
  <jvmserver>DFHWLP</jvmserver>
</provider_pipeline_json>
```

Sustituya DFHWLP por el nombre de JVMSERVER que ha creado al inicio de este procedimiento.

Si ha instalado anteriormente z/OS Connect for CICS 1.0, puede saltarse este paso porque es probable que ya tenga una configuración adecuada.

8. Copie el archivo de configuración en un directorio adecuado en zFS y asegúrese de que los permisos de archivo permiten a la región de CICS leer el archivo. Para obtener información, consulte Archivos de configuración de interconexión.

Si ha instalado anteriormente z/OS Connect for CICS 1.0, puede saltarse este paso porque es probable que ya tenga una configuración adecuada.

9. Cree un recurso PIPELINE. El recurso PIPELINE define la ubicación del archivo de configuración de interconexión en el atributo `CONFIGFILE`.

Si ha instalado anteriormente z/OS Connect for CICS 1.0, puede saltarse este paso porque es probable que ya tenga una configuración adecuada.

10. Opcional: Cree un recurso URIMAP predeterminado para z/OS Connect. Los recursos URIMAP se utilizan para asociar una TRANSACTION y el ID de usuario predeterminado con trabajo de z/OS Connect. Se pueden utilizar uno

o más recursos URIMAP para configurar una política predeterminada para z/OS Connect. Para obtener un ejemplo de configuración URIMAP y más información sobre las opciones de configuración, consulte Configuración de permisos para servicios y API de z/OS.

Nota:

z/OS Connect realiza una autenticación adicional para solicitudes HTTP individuales, así las tareas de aplicación que se ejecutan en CICS están normalmente asociadas a un ID de usuario más específico que el ID de usuario inicial de URIMAP.

Este ID de usuario inicial está sólo en vigor hasta que tiene lugar la autenticación específica de usuario en z/OS Connect. Se recomienda especificar un ID de usuario específico como ZOSCUSER en URIMAP, en lugar de confiar en el ID de usuario de CICS predeterminado que se va a utilizar (normalmente "CICSUSER"), y autorizarlo para utilizar las transacciones de destino.

Resultados

Su instancia de z/OS Connect Enterprise Edition está configurada. Puede probar la configuración básica para z/OS Connect escribiendo esta URL en un navegador web: `https://hostname:portnumber/zosConnect/apim/services`, donde *hostname* es la dirección IP o nombre de host del sistema donde se está ejecutando la región de CICS que aloja z/OS Connect, y *portnumber* es el **httpsPort** que se especifica en el elemento `<httpEndpoint>` del archivo `server.xml`. El navegador web muestra una lista de los servicios instalados: la lista está vacía porque aún no hay servicios instalados.

Si recibe una respuesta HTTP 403 "AuthorizationFailed" en lugar de la lista de servicios esperada, revise la configuración de seguridad del Paso 3. Es probable que el usuario autenticado no esté autorizado para utilizar z/OS Connect.

Qué hacer a continuación

Está preparado para desplegar servicios web JSON o API en z/OS Connect Enterprise Edition. Para obtener información sobre el despliegue de servicios, consulte *Configuring z/OS Connect for a CICS JSON web service*. Para obtener información sobre el despliegue de API, consulte "Uso de APIs de z/OS Connect Enterprise Edition" en la página 80.

Configuración de z/OS Connect for CICS servicio web JSON de CICS

Tras la configuración inicial de z/OS Connect for CICS, puede configurarlo para un servicio web JSON de CICS. Los servicios web de JSON se despliegan en z/OS Connect for CICS de una forma similar a como lo hacen en otros entornos PIPELINE de CICS

Antes de empezar

Debe completar la configuración básica de z/OS Connect antes de iniciar esta tarea. Para z/OS Connect for CICS 1.0, consulte "Configuración de z/OS Connect for CICS 1.0" en la página 71. Para z/OS Connect Enterprise Edition, consulte "Configuración de z/OS Connect Enterprise Edition" en la página 74. También puede necesitar un archivo WSBIND de JSON para cada servicio que desea

desplegar; estos archivos de enlace se pueden generar utilizando el asistente JSON de CICS (los programas de utilidad **DFHLS2JS** y **DFHJS2LS**). Para obtener más información sobre estos programas de utilidad, consulte .

Acerca de esta tarea

Los servicios web de JSON se despliegan en z/OS Connect for CICS de una forma similar a como lo hacen en otros entornos PIPELINE de CICS.

Procedimiento

- Instale un recurso WEBSERVICE que asocia el archivo WSBind al recurso PIPELINE adecuado. Seleccione un nombre significativo para el WEBSERVICE, ya que este nombre también se utiliza como nombre del servicio en z/OS Connect. De manera opcional, puede utilizar el mandato **PERFORM PIPELINE SCAN** para instalar recursos WEBSERVICE. Cualquiera que sea el método que elija, asegúrese de que tiene en cuenta la siguiente información sobre URIMAPs:

El URI, en el que está disponible el servicio en z/OS Connect, se obtiene de una de las siguientes ubicaciones:

- El convenio de denominación predeterminado que utiliza z/OS Connect.
- El URI almacenado en el archivo WSBind, cuando el recurso WEBSERVICE asociado se instala utilizando el mandato **PERFORM PIPELINE SCAN**.
- La configuración específica del servicio en el archivo server.xml de Liberty, si se utiliza el parámetro de configuración **invokeURI**.
- Sólo z/OS Connect Enterprise Edition: El archivo de archivado de aplicación (archivo .aar) para una API que encapsula el servicio.

Si confía en el convenio de denominación predeterminado, el servicio se expone normalmente a través del URI siguiente:

`https://<hostname>:<port>/zosConnect/services/<Service Name>?action=invoke`

En este ejemplo, <Service Name> hace referencia al nombre del servicio (más específicamente, el nombre del recurso WEBSERVICE), y <hostname> y <port> se toman de la configuración del servidor Liberty.

- Opcional: Instale un recurso URIMAP para el servicio z/OS Connect. CICS utiliza un recurso URIMAP para trabajar para el servicio con un ID de transacción específico en CICS, y un ID de usuario inicial. También puede utilizar un único URIMAP para configurar uno o varios servicios. También puede definir un recurso URIMAP predeterminado cuando configure CICS for z/OS Connect. Si no existen ningún URIMAP para que coincida con el URI para un servicio z/OS Connect, las tareas de la aplicación se ejecutan bajo la transacción CJSR, y el ID de usuario predeterminado CICS (normalmente CICSUSER) se utiliza como el ID de usuario inicial. Para obtener más información sobre el rol del ID de usuario inicial, consulte .

Si elige crear un recurso URIMAP, asegúrese de que el ID de usuario asociado tiene autoridad para ejecutar la transacción especificada.

También puede elegir utilizar un URIMAP para asociar el URI a un recurso WEBSERVICE específico. Si un URIMAP está estrechamente enlazado a un WEBSERVICE, el WEBSERVICE de destino se utiliza para todas las solicitudes HTTP que coinciden con URIMAP. Si no se especifica un WEBSERVICE, el WEBSERVICE se selecciona en base al nombre del servicio z/OS Connect. Si el atributo WEBSERVICE del URIMAP queda en blanco, z/OS Connect realiza la correlación él mismo, lo que permite a z/OS Connect Enterprise Edition asociar diferentes servicios a diferentes métodos HTTP dentro de una API.

- Opcional: Actualice el archivo `server.xml`. Algunos patrones de uso pueden necesitar cambios en el archivo de configuración del servidor Liberty, `server.xml`. Normalmente, no es necesario hacer cambios específicos del servicio en el archivo `server.xml`, a menos que necesite un proceso de caso especial para un servicio. Por ejemplo, puede elegir asociar un conjunto específico de interceptores de z/OS Connect con un servicio en el archivo `server.xml`, o exponer un servicio utilizando un URI que no coincide con el convenio de denominación predeterminado de z/OS Connect.

Utilice estos pasos para definir un servicio en el archivo `server.xml`:

1. Asegúrese de que el nombre del servicio deseado no coincide con el nombre de un recurso `WEBSERVICE` en CICS. CICS transmite automáticamente información de configuración a z/OS Connect, y si un servicio definido explícitamente y un `WEBSERVICE` de CICS tienen el mismo nombre, el comportamiento resultante es impredecible.
2. Establezca un valor adecuado para el atributo **`invokeURI`** en el archivo `server.xml` que coincide con el URI utilizado en el `URIMAP`.
3. Enlace el servicio al elemento **`CICSEndpointService`** `serviceRef`.
4. Asegúrese de que existe un `URIMAP` para que coincida el URI para el servicio con el `WEBSERVICE` preciso que se va a utilizar. Si se establece un atributo **`invokeURI`** en el **Paso b.**, el `URIMAP` debe coincidir con ese URI. De lo contrario, el `URIMAP` debe asumir el convenio de denominación de URI predeterminado de z/OS Connect.

El ejemplo siguiente muestra una declaración del servicio z/OS Connect for CICS 1.0 explícita en el archivo `server.xml`:

```
<zoscConnectService invokeURI="/json/myCustomService"
                    serviceName="CICSService1"
                    serviceRef="com.ibm.cics.wlp.zosconnect.CICSEndpointService"/>
```

El ejemplo siguiente muestra la declaración equivalente para z/OS Connect Enterprise Edition en el archivo `server.xml`:

```
<zoscconnect_zoscConnectService invokeURI="/json/myCustomService"
                                serviceName="CICSService1"
                                serviceRef="com.ibm.cics.wlp.zosconnect.CICSEndpointService"/>
```

En ambos ejemplos, el URI `"/json/myCustomService"` se asocia al programa de destinatario de z/OS Connect for CICS, `CICSService1`.

Resultados

Puede probar la configuración para z/OS Connect for CICS escribiendo esta URL en un navegador web: `https://hostname:portnumber/zosConnect/apim/services`. El *nombre de host* es la dirección IP o el nombre de la región de CICS que aloja z/OS Connect for CICS. El *número de puerto* es **`httpsPort`** que se configura en la sección `<httpEndpoint>` del archivo de configuración `server.xml`. El navegador web muestra una lista de los servicios instalados.

El servicio está ahora preparado para invocarse desde un cliente JSON que utiliza el mismo *nombre de host* y *número de puerto*.

Cada servicio instalado tiene una entrada en la lista `zosConnect/apim/services`, por ejemplo:

```
{
  "id": "EXAMPLE",
  "name": "EXAMPLE",
```

```
"url":"https://hostname:portnumber/zosConnect/apim/services/EXAMPLE",
"protocol":"REST",
"description":"CICS Service"
}
```

En este ejemplo, el recurso WEBSERVICE asociado se ha denominado "EXAMPLE", y CICS ha creado esta definición de servicio dinámicamente cuando se ha instalado el recurso de WEBSERVICE EXAMPLE. Puede utilizar un navegador web para visitar <https://hostname:portnumber/zosConnect/apim/services/EXAMPLE> que devuelve un documento con más detalles sobre el servicio, similar al siguiente:

```
{
  "id":"EXAMPLE",
  "name":"EXAMPLE",
  "protocol":"REST",
  "description":"CICS Service",
  "restEndpoints":[
    {
      "name":"EXAMPLE",
      "address":"hostname:portnumber/jsonTests/myExampleService"
    }
  ]
}
```

En este ejemplo, "address" es el URI al que se expone el servicio. Este URI puede derivarse de la información en un archivo WSBInd, desde un atributo **invokeURI**, o desde la convención de denominación de z/OS Connect predeterminada.

Cuando llega una solicitud al servidor JVM de Liberty, está asociada al destinatario de z/OS Connect for CICS que utiliza la información del archivo de configuración `server.xml`. Comienza una nueva tarea de CICS para realizar este trabajo, y está asociada a un recurso WEBSERVICE específico que utiliza la información del recurso URIMAP. El proceso de transformación de datos tiene lugar en el servidor JVM de Liberty y el programa CICS de destino está conectado, como aparece denominado en el archivo WSBInd.

Traslado de los servicios web JSON de Java Pipelines for JSON a z/OS Connect for CICS 1.0 o z/OS Connect Enterprise Edition

El entorno de z/OS Connect es compatible con la tecnología de Java Pipelines for JSON. Puede mover los servicios web JSON desde Java Pipelines for JSON a z/OS Connect for CICS 1.0 o z/OS Connect Enterprise Edition.

Acerca de esta tarea

Por razones de brevedad, en esta sección, se utiliza "z/OS Connect" para hacer referencia a z/OS Connect for CICS 1.0 o z/OS Connect Enterprise Edition.

Volver a desplegar un servicio web JSON desde un Java Pipelines for JSON (como se utiliza con CICS Transaction Server Feature Pack for Mobile Extensions V1.0) a z/OS Connect es un proceso sencillo. Los archivos WSBInd que se utilizan para Java Pipelines for JSON y para z/OS Connect se crean utilizando las mismas herramientas (DFHLS2JS y DFHJS2LS), y son completamente compatibles uno con otro. Si está explorando esta prestación en CICS, puede encontrar ejemplos de servicios web JSON en IBM Redbooks: *Implementing IBM CICS JSON Web Services for Mobile Applications*.

El entorno de z/OS Connect fomenta el uso de SSL entre la aplicación cliente y CICS. Si el entorno de Java Pipelines for JSON existente no utiliza SSL, la conversión a z/OS Connect implica un paso adicional.

Procedimiento

1. Cree la infraestructura de z/OS Connect necesaria utilizando las instrucciones de Configuración de z/OS Connect for CICS. Para parte de esta configuración, seleccione un número de puerto SSL TCP/IP desde el que el servidor Liberty JVM escucha las conexiones entrantes. Dispone de dos opciones a tener en cuenta:
 - a. Seleccionar un número de puerto diferente al número de puerto utilizado por TCPIPService para Java Pipelines for JSON. Esta opción tiene la ventaja de garantizar que se puedan instalar ambos entornos simultáneamente en puertos TCP/IP diferentes. Significa que los programas cliente necesitan actualizarse para apuntar al nuevo servicio JSON. Si el entorno antiguo no utiliza SSL, la conversión a z/OS Connect requiere cambios en el URI, por lo tanto esta opción es más adecuada.
 - b. Seleccionar el mismo número de puerto que el que utiliza TCPIPService para Java Pipelines for JSON. Los números de puerto en los URI utilizados por los programas cliente no necesitan cambiarse, pero no se pueden instalar los dos entornos a la vez. Es posible que el URI deba cambiarse por otros motivos, como el intercambio de HTTP a HTTPS cuando habilita SSL.
2. Despliegue los archivos WSBind para z/OS Connect. Los archivos WSBind existentes son completamente compatibles con z/OS Connect. Siga los pasos para desplegar un nuevo servicio web JSON para z/OS Connect como se describe en Configuración de z/OS Connect para un servicio web JSON de CICS. CICS no permite dos recursos WEBSERVICE instalados con el mismo nombre. Por lo tanto, dispone de dos opciones:
 - a. Descartar el WEBSERVICE original para permitir que el nuevo se instale con el mismo nombre que se estaba utilizando anteriormente.
 - b. Cambiar el nombre del WEBSERVICE nuevo para evitar un conflicto.
3. Si ha personalizado el proceso de Java Pipelines for JSON a través del uso de programas de manejador de PIPELINE, piense si esa personalización aún es necesaria. Si es necesaria, cree programas de interceptor de z/OS Connect con funciones equivalentes y despléguelas como interceptores globales. Para obtener más información sobre los interceptores de z/OS Connect, consulte Defining z/OS Connect interceptors.

Resultados

Ahora está preparado para probar los nuevos servicios web JSON de z/OS Connect. Si ha desplegado los servicios utilizando el mismo nombre de puerto que se utilizaba en Java Pipelines for JSON, no son necesarios cambios en el cliente (a menos que la configuración de seguridad haya cambiado, como cuando se habilita SSL). Si ha cambiado el número de puerto o URI para el servicio, se necesita cambiar el cliente.

Uso de APIs de z/OS Connect Enterprise Edition

Una característica principal de z/OS Connect Enterprise Edition es la capacidad de componer APIs de JSON desde uno o varios servicios JSON. Para trabajar con APIs, debe instalar el componente Editor de API de z/OS Connect Enterprise Edition.

Para obtener detalles sobre el Editor de API, consulte z/OS Connect Enterprise Edition documentation in IBM Knowledge Center.

Hay algunas pequeñas diferencias en el despliegue de APIs para z/OS Connect Enterprise Edition for CICS, comparado con el despliegue de APIs para una

instalación autónoma de z/OS Connect Enterprise Edition. Las siguientes consideraciones se aplican al despliegue de APIs para CICS:

- No puede adoptar un archivo WSBInd existente para utilizarlo con una API. El Editor de API requiere uno o varios archivos de Service Archive Resource (SAR) como parte de su entrada. Este archivo se crea con los asistentes BAQLS2JS o BAQJS2LS distribuidos con z/OS Connect Enterprise Edition. Si desea crear APIs, empiece generando los archivos WSBInd y SAR necesarios utilizando los asistentes que se proporcionan con z/OS Connect Enterprise Edition.

Debe obtener el parámetro **SERVICE-NAME** en BAQLS2JS o BAQJS2LS para hacer coincidir el nombre del recurso WEBSERVICE que se utilizará en CICS. CICS confía en una coincidencia uno a uno entre el nombre del servicio en z/OS Connect y el nombre de WEBSERVICE en CICS.

- Se necesita una definición de recurso WEBSERVICE en CICS que coincide con el nombre del servicio. WEBSERVICE encapsula el archivo WSBInd para el servicio.

-

También se puede proporcionar una definición de recurso URIMAP en CICS. Si se utiliza URIMAP, CICS asocia automáticamente el trabajo de la API a un ID de transacción y establece un ID de usuario inicial adecuado.

- El Editor de API crea un archivo Recurso de archivado de aplicación (AAR) como salida. Debe desplegarse en el servidor Liberty utilizando los mecanismos de despliegue proporcionados por z/OS Connect Enterprise Edition. Es posible que necesite crear un directorio adecuado para las API en el área de trabajo del servidor Liberty. Las APIs se despliegan en el directorio /resources/zosconnect/apis en el área de trabajo Liberty. Si este directorio no existe, créelo. Un ejemplo de uso de apideploy es el siguiente:

```
apideploy -deploy -a /tmp/ExampleAPI.aar -p
/u/userid/APPLID/JVMSEVER/wlp/usr/servers/defaultServer/resources/zosconnect/apis
```

En el ejemplo dado, /tmp/ExampleAPI.aar es la ubicación del Recurso de archivado de aplicación que se está desplegando y /u/userid/APPLID/JVMSEVER/wlp/usr/servers/ es el valor de la propiedad WLP_OUTPUT_DIR en el archivo de configuración JVMSEVER. Este ejemplo da como resultado en apideploy la actualización de la configuración del servidor Liberty que está utilizando CICS.

Si sigue las consideraciones indicadas arriba, la API y sus servicios de componente se despliegan en CICS. CICS se configura con recursos URIMAP y WEBSERVICE adecuados, y el archivo AAR se despliega en el área de trabajo de configuración de Liberty. El archivo de configuración de Liberty principal, server.xml, contiene una entrada para cada servicio de componente que se despliega.

Archivos de configuración de interconexión

La configuración de una interconexión utilizada para manejar una solicitud de servicio web se especifica en un documento XML, conocido como *archivo de configuración de interconexión*.

El archivo de configuración de interconexión se almacena en el sistema de archivos de z/OS UNIX System Services y su nombre se especifica en el atributo CONFIGFILE de una definición de recurso PIPELINE. Utilice un editor XML o un editor de texto adecuado para trabajar con los archivos de configuración de interconexión. Los esquemas XML para los archivos de configuración de interconexión están en el directorio /usr/lpp/cicsts/cicsts54/schemas/pipeline/ (donde /usr/lpp/cicsts/cicsts54 es el directorio de instalación predeterminado para los archivos CICS en z/OS UNIX). Cuando trabaja con archivos de

configuración, asegúrese de que la codificación del conjunto de caracteres es UTF-8. Si importa un archivo de configuración existente codificado en EBCDIC, se convierte automáticamente en UTF-8.

Cuando CICS procesa una solicitud de servicio web, utiliza una interconexión de uno o varios manejadores de mensajes para manejar la solicitud. Una interconexión se configura para proporcionar aspectos del entorno de ejecución que se aplican a diferentes categorías de aplicaciones, como soporte para la seguridad de servicios web y transacciones de servicios web. Normalmente, una región de CICS que tiene un gran número de aplicaciones de proveedor de servicios o de solicitante de servicio necesita varias configuraciones de interconexión diferentes. Sin embargo, donde aplicaciones diferentes tienen requisitos similares, pueden compartir la misma configuración de interconexión.

Nota: Al utilizar CICS Explorer para crear un archivo de configuración PIPELINE nuevo como parte de un paquete, no debe haber un archivo de configuración con el mismo nombre en la raíz del paquete.

Hay dos tipos de configuraciones de interconexión: una describe la configuración de una interconexión de proveedor de servicios; la otra describe una interconexión de solicitante de servicio. Cada una está definida por su propio esquema y tiene un elemento raíz diferente.

Interconexión	Esquema	Elemento raíz
Proveedor de servicios	Provider.xsd	<provider_pipeline>
Solicitante de servicio	Requester.xsd	<requester_pipeline>

Aunque muchos de los elementos XML utilizados son comunes para ambos tipos de configuración de interconexión, otros se utilizan únicamente uno u otro, por lo que no puede utilizar el mismo archivo de configuración para el proveedor y el solicitante.

Restricción: Los nombres de elemento cualificados por el espacio de nombres no se soportan en el archivo de configuración de interconexión.

Los elementos <provider_pipeline> y <requester_pipeline> tienen los siguientes subelementos inmediatos:

- Un elemento <service>, que especifica los manejadores de mensajes que se están invocando para cada solicitud. Este elemento es obligatorio cuando se utiliza en el elemento <provider_pipeline>, y opcional en el elemento <requester_pipeline>.
- Un elemento <transport> opcional, que especifica los manejadores de mensajes que se seleccionan en el tiempo de ejecución, basándose en los recursos que se están utilizando para el transporte de mensajes.
- Sólo para <provider_pipeline>, un elemento <apphandler> opcional, que se utiliza para especificar manejadores de aplicación conectados por canal.
- Sólo para <provider_pipeline>, un elemento <apphandler_class> opcional, que se utiliza para especificar un manejador de aplicación Axis2.
- Un elemento <service_parameter_list> opcional, que contiene los parámetros que están disponibles para los manejadores de mensajes en la interconexión.

Ciertos elementos pueden tener atributos asociados a ellos. Cada valor de atributo debe estar entre comillas para dar un documento XML válido .

Asociado al archivo de configuración de interconexión hay un recurso PIPELINE. Los atributos incluyen CONFIGFILE, que especifica el nombre del archivo de configuración de interconexión en z/OS UNIX. Cuando instala una definición PIPELINE, CICS lee la información que necesita para configurar la interconexión desde el archivo.

CICS proporciona archivos de configuración de ejemplo que puede utilizar como base para el desarrollo de sus propios archivos de configuración. Se proporcionan en la biblioteca /usr/lpp/cicsts/cicsts54/samples/pipelines.

basicsoap11provider.xml

Una definición de interconexión de proveedor de servicios que utiliza el protocolo SOAP 1.1 para una interconexión que no soporta Java. La interconexión utiliza el manejador de mensajes <cics_soap_1.1_handler> y se utiliza cuando la aplicación CICS se ha desplegado utilizando el asistente de servicios web de CICS.

basicsoap11requester.xml

Una definición de interconexión de solicitante de servicio que utiliza el protocolo SOAP 1.1 para una interconexión que no soporta Java. La interconexión utiliza el manejador de mensajes <cics_soap_1.1_handler> y se utiliza cuando la aplicación CICS se ha desplegado utilizando el asistente de servicios web de CICS.

basicsoap11javaprovider.xml

Una definición de interconexión de proveedor de servicios que utiliza el protocolo SOAP 1.1 para una interconexión que soporta Java. La interconexión utiliza el manejador de mensajes <cics_soap_1.1_handler_java> y se utiliza cuando la aplicación se ha desplegado utilizando el asistente de servicios web de CICS. Esta configuración contiene el elemento <jvmserver>. Este manejador de mensajes debe editarse para especificar el servidor JVM adecuado antes de que se pueda utilizar la configuración.

basicsoap11javarequester.xml

Una definición de interconexión de solicitante de servicio que utiliza el protocolo SOAP 1.1 para una interconexión que soporta Java. La interconexión utiliza el manejador de mensajes <cics_soap_1.1_handler_java> y se utiliza cuando la aplicación se ha desplegado utilizando el asistente de servicios web de CICS. Esta configuración contiene el elemento <jvmserver>. Este manejador de mensajes debe editarse para especificar el servidor JVM adecuado antes de que se pueda utilizar la configuración.

jsonjavaprovider.xml

Una definición de interconexión de proveedor de servicios que utiliza el formato de mensaje JSON para una interconexión que soporta Java. La interconexión utiliza el manejador de mensajes <cics_json_handler_java> y se utiliza cuando la aplicación CICS se ha desplegado utilizando el asistente de JSON de CICS. Esta configuración contiene el elemento <jvmserver>. Este manejador de mensajes debe editarse para especificar el servidor JVM adecuado antes de que se pueda utilizar la configuración.

jsonzosconnectprovider.xml

Una definición de interconexión para un servicio web JSON que se despliega en una PIPELINE configurada para z/OS Connect for CICS. La interconexión utiliza el manejador de mensajes <provider_pipeline_json>. Esta configuración contiene el elemento <jvmserver>. Este manejador de

mensajes debe editarse para especificar el servidor JVM adecuado antes de que se pueda utilizar la configuración.

kerberosprovider.xml

Una definición de interconexión de proveedor de servicios que añade información de configuración para el soporte Kerberos a basicsoap11provider.xml.

samlprovider.xml

Una definición de interconexión de proveedor de servicios que añade información de configuración para el soporte SAML a basicsoap11provider.xml.

samlrequester.xml

Una definición de interconexión de proveedor de servicios que añade información de configuración para el soporte SAML a basicsoap11requester.xml.

propagatesamlprovider.xml

Una definición de interconexión de proveedor de servicios que añade información de configuración para el soporte SAML con propagación de información SAML a través de una transacción CICS a basicsoap11provider.xml.

propagatesamlrequester.xml

Una definición de interconexión de solicitante de servicio que añade información de configuración para el soporte SAML con propagación de información SAML a través de una transacción CICS a basicsoap11requester.xml.

wsatprovider.xml

Una definición de interconexión que añade información de configuración para transacciones de servicios web a basicsoap11provider.xml.

wsatrequester.xml

Una definición de interconexión que añade información de configuración para transacciones de servicios web a basicsoap11requester.xml.

Archivo de configuración de interconexión de proveedor de ejemplo (manejador de aplicación JSON)

Es un sencillo ejemplo de un archivo de configuración para una interconexión de proveedor de servicios que utiliza el elemento `<cics_json_handler_java>`:

```
<?xml version="1.0" encoding="EBCDIC-CP-US"?>
<provider_pipeline xmlns="http://www.ibm.com/software/http/cics/pipeline">
  <service>
    <terminal_handler>
      <cics_json_handler_java>
        <jvmserver>DFHAXIS</jvmserver>
        <repository>/usr/lpp/cicsts/cicsts52/lib/pipeline/repository</repository>
      </cics_json_handler_java>
    </terminal_handler>
  </service>
  <apphandler_class>com.ibm.cicsts.axis2.CICSAXIS2ApplicationHandler</apphandler_class>
</provider_pipeline>
```

La interconexión contiene sólo un manejador de mensajes. El manejador enlaza con el programa DFHJSON.

- El elemento `<provider_pipeline>` es el elemento raíz del archivo de configuración de interconexión para una interconexión de proveedor de servicios.
- El elemento `<service>` especifica los manejadores de mensajes que se invocan para cada solicitud. En este ejemplo, sólo hay un manejador de mensajes.
- El elemento `<terminal_handler>` contiene la definición de manejador de mensajes de terminal de la interconexión.
- El elemento `<cics_json_handler_java>` indica que la interconexión es una interconexión basada en Java y el manejador de servicios de la interconexión es un manejador de mensajes que soporta mensajes JSON.
- El elemento `<apphandler>` especifica el nombre del manejador de aplicación con el que está conectado el controlador de terminal de la interconexión de forma predeterminada. En este caso, el programa es DFHJSON, que es el programa proporcionado por CICS para aplicaciones desplegadas con el asistente de JSON de CICS.

Archivo de configuración de interconexión de proveedor de ejemplo (manejador de aplicación conectado por canal)

Es un sencillo ejemplo de un archivo de configuración para una interconexión de proveedor de servicios que utiliza el elemento `<cics_soap_1.1_handler>`:

```
<?xml version="1.0" encoding="EBCDIC-CP-US"?>
<provider_pipeline
  xmlns="http://www.ibm.com/software/hcp/cics/pipeline"
  <service>
    <terminal_handler>
      <cics_soap_1.1_handler/>
    </terminal_handler>
  </service>
  <apphandler>DFHPITP</apphandler>
</provider_pipeline>
```

La interconexión contiene sólo un manejador de mensajes. El manejador enlaza con el programa DFHPITP.

- El elemento `<provider_pipeline>` es el elemento raíz del archivo de configuración de interconexión para una interconexión de proveedor de servicios.
- El elemento `<service>` especifica los manejadores de mensajes que se invocan para cada solicitud. En este ejemplo, sólo hay un manejador de mensajes.
- El elemento `<terminal_handler>` contiene la definición de manejador de mensajes de terminal de la interconexión.
- El elemento `<cics_soap_1.1_handler>` indica que la interconexión no es una interconexión basada en Java y el manejador de terminal de la interconexión es un manejador de mensajes que soporta mensajes SOAP 1.1.
- El elemento `<apphandler>` especifica el nombre del manejador de aplicación con el que está conectado el controlador de terminal de la interconexión de forma predeterminada. En este caso, el programa es DFHPITP, que es el programa proporcionado por CICS para aplicaciones desplegadas con el asistente de servicios web de CICS.

Archivo de configuración de interconexión de proveedor de ejemplo (manejador de aplicación Axis2)

Es un sencillo ejemplo de un archivo de configuración para una interconexión de proveedor de servicios que utiliza el elemento `<cics_soap_1.1_handler_java>`:

```
<?xml version="1.0" encoding="EBCDIC-CP-US"?>
<provider_pipeline
  xmlns="http://www.ibm.com/software/http/cics/pipeline"
  <service>
    <terminal_handler>
      <cics_soap_1.1_handler_java>
        <jvmserver>DFHAXIS</jvmserver>
      </cics_soap_1.1_handler_java>
    </terminal_handler>
  </service>
  <apphandler_class>com.ibm.cicsts.axis2.CICSAxis2ApplicationHandler</apphandler_class>
</provider_pipeline>
```

La interconexión contiene sólo un manejador de mensajes. El manejador enlaza con el programa DFHPITP.

- El elemento `<provider_pipeline>` es el elemento raíz del archivo de configuración de interconexión para una interconexión de proveedor de servicios.
- El elemento `<service>` especifica los manejadores de mensajes que se invocan para cada solicitud. En este ejemplo, sólo hay un manejador de mensajes.
- El elemento `<terminal_handler>` contiene la definición de manejador de mensajes de terminal de la interconexión.
- El elemento `<cics_soap_1.1_handler_java>` indica que la interconexión es una interconexión basada en Java y el manejador de servicios de la interconexión es un manejador de mensajes que soporta mensajes SOAP 1.1.
- El elemento `<apphandler_class>` especifica el manejador de aplicación Axis2 proporcionado.

Archivo de configuración de interconexión de solicitante de ejemplo

Es un sencillo ejemplo de un archivo de configuración para una interconexión de solicitante de servicio que utiliza el elemento `<cics_soap_1.2_handler_java>` con soporte Axis2 MTOM/XOP:

```
<?xml version="1.0" encoding="EBCDIC-CP-US"?>
<requester_pipeline
  xmlns="http://www.ibm.com/software/http/cics/pipeline">
  <service>
    <service_handler_list>
      <cics_soap_1.2_handler_java>
        <jvmserver>JVMSERV1</jvmserver>
        <mtom>
      </cics_soap_1.2_handler_java>
    </service_handler_list>
  </service>
</requester_pipeline>
```

La interconexión contiene sólo un manejador de mensajes.

- El elemento `<requester_pipeline>` es el elemento raíz del archivo de configuración de interconexión para una interconexión de solicitante de servicio.
- El elemento `<service>` especifica los manejadores de mensajes que se invocan para cada solicitud. En este ejemplo, sólo hay un manejador de mensajes.
- `<service_handler_list>` especifica una lista de manejadores de mensajes que se invocan para cada solicitud.
- El elemento `<cics_soap_1.2_handler_java>` indica que la interconexión soporta Java y el manejador de servicios de la interconexión es un manejador de mensajes que soporta mensajes SOAP 1.2.
- El elemento `<jvmserver>` especifica el servidor JVM que se va a utilizar.

- El elemento <mtom/> especifica que los documentos XOP de salida se empaquetan en mensajes MTOM y se envían. De forma predeterminada, los mensajes MTOM de entrada se aceptan y desempaquetan para interconexiones basadas en Java.

Archivo de configuración de interconexión de proveedor de ejemplo para un servicio web de JSON z/OS Connect for CICS

Es un sencillo ejemplo de un archivo de configuración para una interconexión de proveedor de servicios que utiliza el elemento <provider_pipeline_json>. Como se proporciona un atributo java_parser="N0", utiliza el analizador JSON que no es de Java:

```
<?xml version="1.0" encoding="EBCDIC-CP-US"?>
<provider_pipeline_json java_parser="N0"
  xmlns="http://www.ibm.com/software/http/cics/pipeline">
  <jvmserver>DFHWLP</jvmserver>
</provider_pipeline_json>
```

El elemento <provider_pipeline_json> difiere del elemento <provider_pipeline> en que no se pueden definir programas de manejador.

- El elemento <provider_pipeline_json> es el elemento raíz del archivo de configuración de interconexión para una interconexión de proveedor de servicios web de JSON z/OS Connect for CICS.
- El atributo java_parser="N0" especifica que se utiliza el analizador JSON que no es de Java.
- El elemento <jvmserver> especifica el servidor JVM que se va a utilizar.

Nota: Un intento para iniciar una interconexión <provider_pipeline_json> utilizando cualquier cosa que no sea z/OS Connect for CICS da como resultado un error.

Archivo de configuración de interconexión de proveedor de ejemplo para un análisis JSON que no es de Java

```
<?xml version="1.0" encoding="EBCDIC-CP-US"?>
<provider_pipeline
  xmlns="http://www.ibm.com/software/http/cics/pipeline">
  <service>
    <terminal_handler>
      <handler>
        <program>DFHPIJT</program><handler_parameter_list/>
      </handler>
    </terminal_handler>
  </service>
</provider_pipeline>
```

La interconexión contiene sólo un manejador de mensajes.

- El elemento <provider_pipeline> es el elemento raíz del archivo de configuración de interconexión para una interconexión de proveedor de servicios.
- El elemento <service> especifica los manejadores de mensajes que se invocan para cada solicitud. En este ejemplo, solo hay un manejador de mensajes.
- El elemento <terminal_handler> contiene la definición de manejador de mensajes de terminal de la interconexión.
- El elemento <handler> especifica los detalles del manejador.
- El elemento <program> especifica el programa que se va a invocar. DFHPIJT es el manejador proporcionado por CICS para un proceso JSON que no es Java.

Manejadores relacionados con el transporte

En el archivo de configuración para cada interconexión, puede especificar más de un conjunto de manejadores de mensaje. En tiempo de ejecución, CICS selecciona los manejadores de mensaje que se invocan, basándose en los recursos que se están utilizando para el transporte de mensajes.

En un proveedor de servicios, y en un solicitante de servicio, puede especificar que algunos manejadores de mensajes deben invocarse sólo cuando un transporte particular (HTTP o WebSphere MQ) está en uso. Por ejemplo, tenga en cuenta un servicio web que pone a disposición de sus empleados. Aquellos que trabajan en una ubicación de la compañía acceden al servicio utilizando el transporte WebSphere MQ en una red interna segura; sin embargo, los empleados que trabajan en una ubicación de business partner acceden al servicio utilizando el transporte HTTP a través de Internet. En esta situación, es posible que desee utilizar los manejadores de mensaje para cifrar partes del mensaje cuando se utiliza el transporte HTTP, debido a la naturaleza sensible de la información.

En un proveedor de servicios, los mensajes de entrada están asociados a un recurso con nombre (un TCPIP SERVICE para el transporte HTTP, una QUEUE para el transporte MQ). Puede especificar que algunos manejadores de mensajes deben invocarse únicamente cuando se utiliza un recurso particular para una solicitud de entrada.

Para hacer esto posible, los manejadores de mensaje se especifican en dos partes distintas del archivo de configuración de interconexión:

La sección de servicio

Especifica los manejadores de mensaje que se invocan cada vez que se ejecuta la interconexión.

La sección de transporte

Especifica los manejadores de mensaje que se pueden o no invocar, dependiendo de los recursos de transporte que están en uso.

Recuerde: En tiempo de ejecución, un manejador de mensajes puede decidir acortar la ejecución de la interconexión. Por lo tanto, aunque CICS decida que debe invocarse un manejador de mensajes particular basándose en lo que hay en el archivo de configuración de interconexión, el manejador de mensajes anterior puede invalidar la decisión.

Los manejadores de mensajes especificados en la sección de transporte (los *manejadores relacionados con el transporte*) están organizados en varias listas. En tiempo de ejecución, CICS selecciona los manejadores en una de estas listas de ejecución, basándose en los recursos de transporte que están en uso. Si más de una lista coincide con los recursos de transporte que se están utilizando, CICS utiliza la lista que es más selectiva. Las listas que se usan en las interconexiones de proveedor de servicios y solicitante de servicios son:

<default_transport_handler_list>

Esta es la lista de manejadores relacionados con el transporte menos selectiva; los manejadores especificados en esta lista se invocan cuando ninguna de las listas siguientes coincide con los recursos de transporte que se están utilizando.

<default_http_transport_handler_list>

En una interconexión de solicitante de servicio, los manejadores de esta lista se invocan cuando el transporte HTTP está en uso.

En una interconexión de proveedor de servicios, los manejadores de esta lista se invocan cuando el transporte HTTP está en uso, y ninguna `<named_transport_entry>` nombra el TCIPSERVICE para la conexión TCP/IP.

`<default_mq_transport_handler_list>`

En una interconexión de solicitante de servicio, los manejadores de esta lista se invocan cuando el WebSphere MQ está en uso.

En una interconexión de proveedor de servicios, los manejadores de la lista se invocan cuando el transporte WebSphere MQ está en uso, y ninguna `<named_transport_entry>` nombra la cola de mensajes en la que se reciben los mensajes de entrada.

La lista de manejadores de mensajes siguiente se utiliza sólo en el archivo de configuración para una interconexión de proveedor de servicios:

`<named_transport_entry>`

Igual que con la lista de manejadores, `<named_transport_entry>` especifica el nombre de un recurso y el tipo de transporte.

- Para el transporte HTTP, los manejadores de la lista se invocan cuando el nombre de recurso coincide con el nombre de TCIPSERVICE para la conexión TCP/IP de entrada.
- Para el transporte WebSphere MQ, los manejadores de la lista se invocan cuando el nombre de recurso coincide con el nombre de la cola de mensajes que recibe el mensaje de entrada.

Ejemplo

Este es un ejemplo de un elemento `<transport>` del archivo de configuración de interconexión para una interconexión de proveedor de servicios

```
<transport>

  <!-- HANDLER1 and HANDLER2 are the default transport handlers -->
  <default_transport_handler_list>
    <handler><program>HANDLER1</program><handler_parameter_list/></handler>
    <handler><program>HANDLER2</program><handler_parameter_list/></handler>
  </default_transport_handler_list>

  <!-- HANDLER3 overrides defaults for MQ transport -->
  <default_mq_transport_handler_list>
    <handler><program>HANDLER3</program><handler_parameter_list/></handler>
  </default_mq_transport_handler_list>

  <!-- HANDLER4 overrides defaults for http transport with TCIPSERVICE(WS00) -->
  <named_transport_entry type="http">
    <name>WS00</name>
    <transport_handler_list>
      <handler><program>HANDLER4</program><handler_parameter_list/></handler>
    </transport_handler_list>
  </named_transport_entry>

</transport>
```

El efecto de esta definición es éste:

- La `<default_mq_transport_handler_list>` asegura que los mensajes que utilizan el transporte MQ se procesan mediante el manejador HANDLER3.
- La `<named_transport_entry>` asegura que los mensajes que utilizan la conexión TCP/IP asociada a TCIPSERVICE(WS00) se procesan mediante el manejador HANDLER4.

- La `<default_transport_handler_list>` garantiza que todos los mensajes que quedan, es decir, los que utilizan el transporte HTTP, pero no TCPISERVICE(WS00), se procesan mediante los manejadores HANDLER1 y HANDLER2.

Recuerde: Los manejadores especificados en la sección de servicio de la definición de interconexión se invocarán además de los especificados en la sección de transporte.

Definición de interconexión para un proveedor de servicios

Los manejadores de mensajes se definen en un documento XML, que se almacena en z/OS UNIX. El nombre del archivo que contiene el documento se especifica en el atributo CFGFILE de una definición PIPELINE.

El elemento raíz del documento de configuración de interconexión es el elemento `<provider_pipeline>`. La estructura de alto nivel del documento se muestra en Figura 20 en la página 91.

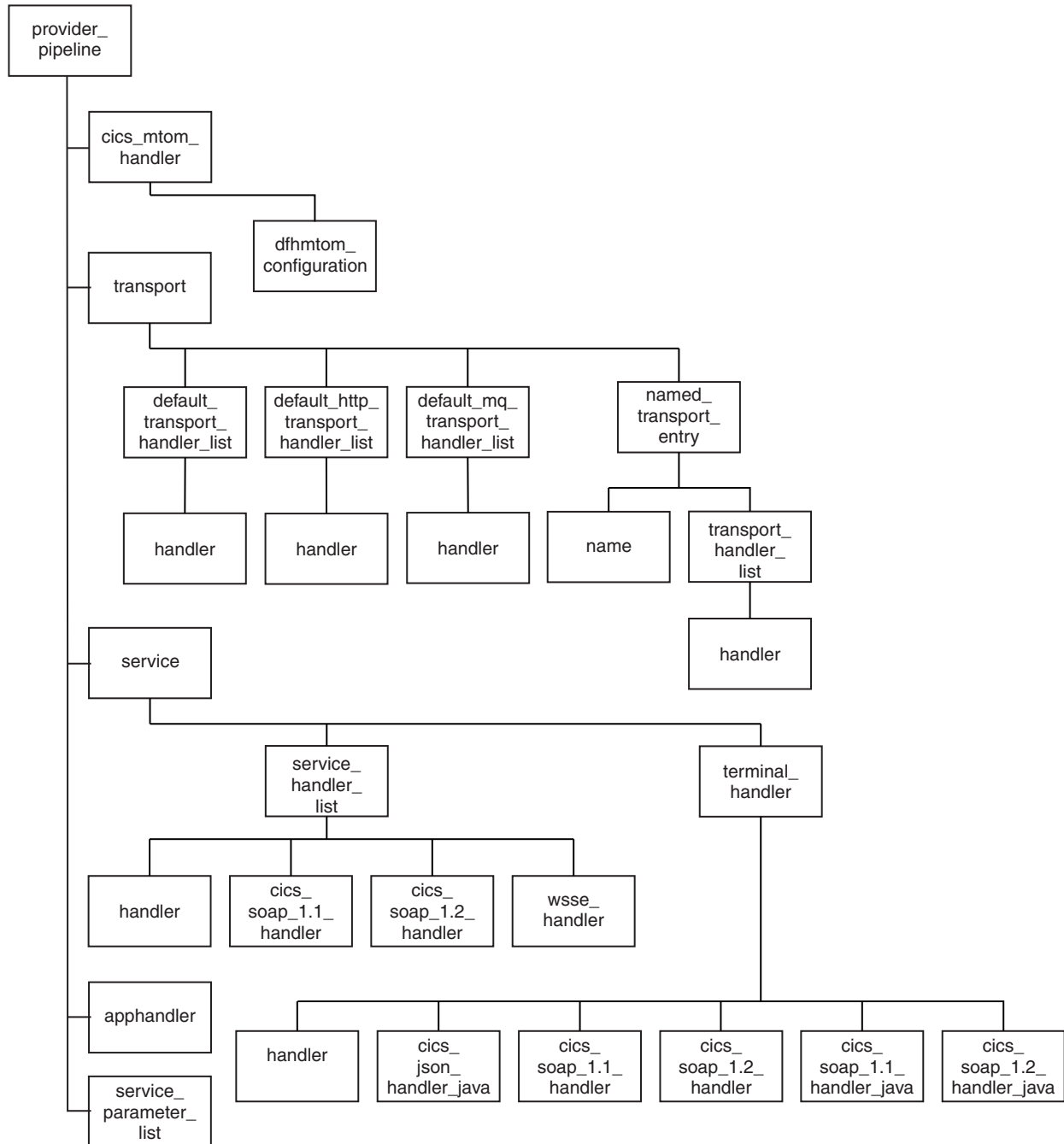


Figura 20. Estructura de la definición de interconexión para un proveedor de servicios.

Nota: Para simplificar la figura, los elementos hijo de los elementos <handler>, <cics_json_handler_java>, <cics_soap_1.1_handler>, <cics_soap_1.2_handler>, <cics_soap_1.1_handler_java> y <cics_soap_1.2_handler_java> no se muestran.

Definición de interconexión para un solicitante de servicio

Los manejadores de mensajes se definen en un documento XML, que está almacenado en z/OS UNIX. El nombre del archivo que contiene el documento se especifica en el atributo CFGFILE de una definición PIPELINE.

El elemento raíz del documento de configuración de interconexión es el elemento `<requester_pipeline>`. La estructura de alto nivel del documento se muestra en Figura 21.

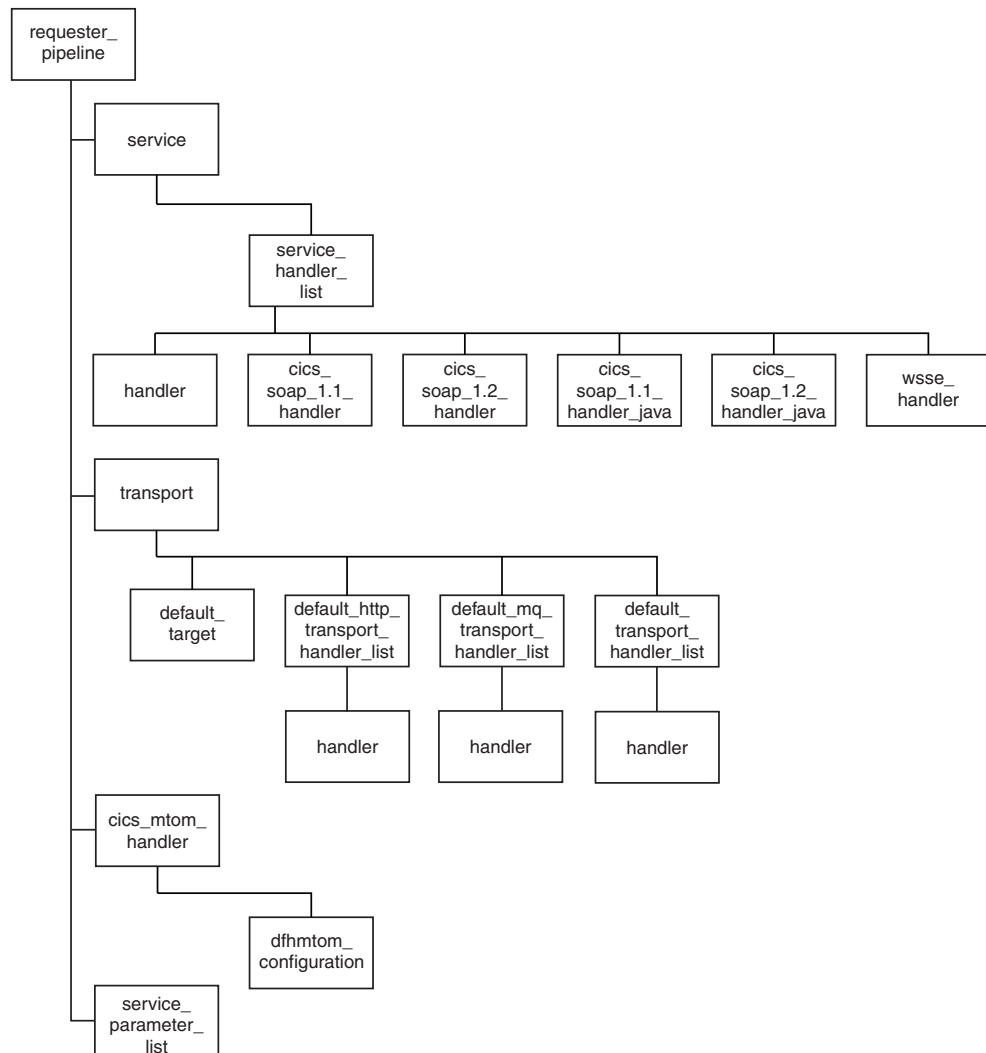


Figura 21. Estructura de la definición de interconexión para un solicitante de servicio.

Nota: Para simplificar la figura, los elementos hijo de los elementos `<handler>`, `<cics_soap_1.1_handler>`, `<cics_soap_1.2_handler>`, `<cics_soap_1.1_handler_java>` y `<cics_soap_1.2_handler_java>` no se muestran.

Elementos utilizados únicamente en proveedores de servicios

Algunos de los elementos XML utilizados en un archivo de configuración de interconexión se aplican únicamente a interconexiones de proveedor de servicios.

Manejadores de aplicación:

Un manejador de aplicación es un programa CICS al que se enlaza el controlador de terminal de una interconexión de proveedor de servicio SOAP en el tiempo de ejecución.

Los manejadores de aplicación se utilizan en las interconexiones de modalidad de proveedor en las que el controlador de terminal es uno de los manejadores de mensajes SOAP proporcionados. Esta situación se produce cuando el elemento `<terminal_handler>` contiene un elemento `<cics_soap_1.1_handler>`, `<cics_soap_1.2_handler>`, `<cics_soap_1.1_handler_java>` o `<cics_soap_1.2_handler_java>`.

El manejador de aplicación es responsable de procesar el cuerpo de una solicitud SOAP y de generar una respuesta utilizando los datos devueltos. El manejador de aplicación puede invocar a otros programas para completar este proceso. Normalmente, los manejadores de aplicación actúan como una capa de presentación de propósito general alrededor de una o varias aplicaciones empresariales. Es responsable de la correlación de XML en un formato que puede utilizar una aplicación, de conectar esa aplicación y de generar después una respuesta utilizando los datos devueltos.

CICS puede conectar un manejador de aplicación de dos formas. El mecanismo típico implica un canal y contenedores de control; el otro método implica enlaces Java para Axis2.

Los manejadores de aplicación conectados por canal se especifican en el elemento `<apphandler>` del elemento `<provider_pipeline>`. En el tiempo de ejecución, el contenedor DFHWS-APPHANDLER se llena con el contenido de `<apphandler>`. Sin embargo, el contenedor DFHWS-APPHANDLER se puede actualizar dinámicamente por medio de cualquiera de los manejadores de mensajes. Por lo tanto, el programa con el que se enlaza en el tiempo de ejecución puede ser diferente al programa especificado en el elemento `<apphandler>`. Los siguientes manejadores de aplicación se pueden especificar en el elemento `<apphandler>` o el contenedor DFHWS-APPHANDLER:

- El manejador de aplicación SOAP conectado por canal proporcionado, DFHPITP. Para obtener más información sobre los manejadores de aplicación conectados por canal, consulte “Manejadores de aplicación conectados por canal” en la página 135
- Su propio manejador de aplicación conectado por canal. Este manejador de aplicación se puede escribir en lenguajes distintos a Java. Para obtener más información sobre los contenedores de control que se pueden utilizar en el manejador de aplicación conectado por canal, consulte “Contenedores de control” en la página 152.
- Su propio manejador de aplicación Java para interconexiones basadas en Java, que implementa la interfaz de Java `ApplicationHandler` y que está conectado a la interconexión utilizando Axis2 `MessageContext`. Para obtener más información sobre la interfaz de Java `ApplicationHandler`, consulte `JCICS class reference`.

Para utilizar un manejador de aplicación que utiliza enlaces Java para Axis2, debe especificar el elemento `<apphandler_class>` del elemento `<provider_pipeline>`. Los manejadores de aplicación Axis2 también requieren que debe existir un servidor JVM para la interconexión de servicios web y el manejador de aplicación en el que ejecutarse y que el controlador de terminal de la interconexión de servicios web debe estar en el manejador de mensajes `<cics_soap_1.1_handler_java>` o `<cics_soap_1.2_handler_java>`. Para utilizar el manejador de aplicación Axis2 proporcionado, debe especificar `com.ibm.cicsts.axis2.CICSAXIS2ApplicationHandler` en el elemento `<apphandler_class>`, sin embargo, puede especificar su propia clase de manejador de aplicación Axis2. En el tiempo de ejecución, el contenedor DFHWS-APPHANDLER se llena con el contenido de `<apphandler_class>`.

Para aplicaciones de servicio web desplegadas utilizando el asistente de servicios web CICS, debe especificar DFHPITP o el manejador de aplicación que utiliza DFHPITP en el elemento `<apphandler>`, o especificar `com.ibm.cicsts.axis2.CICSAxis2ApplicationHandler` en el elemento `<apphandler_class>`. Para obtener más información sobre el asistente de servicios web CICS, consulte The CICS web services assistant.

También es posible desplegar aplicaciones Axis2 como servicios web de modalidad de proveedor en CICS utilizando el estilo Axis2 del despliegue de servicio web. Para obtener más información, consulte el apartado Deploying a Java provider-mode web service in an Axis2 JVM server.

Elemento `<apphandler_class>`:

Especifica que el controlador de terminal de la interconexión enlaza con un manejador de aplicación Axis2.

El elemento `<apphandler_class>` se utiliza para especificar un manejador de aplicación Axis2 cuando el elemento `<terminal_handler>` contiene un elemento `<cics_json_handler_java>`, `<cics_soap_1.1_handler_java>`, o `<cics_soap_1.2_handler_java>`. Para utilizar el manejador de aplicación Axis2 proporcionado, especifique `com.ibm.cicsts.axis2.CICSAxis2ApplicationHandler` en el elemento `<apphandler_class>`. Cuando se utilizan los manejadores SOAP de CICS, también puede especificar su propia clase de manejador de aplicación Axis2.

Como alternativa, puede especificar el elemento `<apphandler>` en el archivo de configuración de interconexión si desea utilizar un manejador de aplicación conectado por canal, para obtener más información, consulte el elemento `<apphandler>`. Sin embargo, no debe especificar los elementos `<apphandler_class>` y `<apphandler>` en el mismo archivo de configuración de interconexión.

Nota: No debe utilizar el elemento `<apphandler>` con el elemento `<cics_json_handler_java>`.

No debe utilizar el elemento `<apphandler_class>` si el elemento `<terminal_handler>` contiene un elemento `<cics_soap_1.1_handler>` o `<cics_soap_1.2_handler>`.

Para obtener más información sobre los manejadores de aplicación, consulte “Manejadores de aplicación” en la página 92.

Se usa en lo siguiente:

- Proveedor de servicios

Contenido en:

- Elemento `<provider_pipeline>`

Ejemplo

```
<apphandler_class>com.ibm.cicsts.axis2.CICSAxis2ApplicationHandler</apphandler_class>
```

Elemento `<named_transport_entry>`:

Contiene una lista de manejadores que se invocan cuando el proveedor de servicios está utilizando un recurso de transporte con nombre.

- Para el transporte WebSphere MQ, el recurso con nombre es la cola de entrada local donde se recibe la solicitud.
- Para el transporte HTTP, el recurso es el TCPIPService que define el puerto en el que se ha recibido la solicitud.

Se usa en lo siguiente:

- Proveedor de servicios

Contenido en:

<transporte>

Atributos:

Nombre	Descripción
tipo	El mecanismo de transporte al que se asocia el recurso con nombre: wmq El recurso con nombre es una cola http El recurso con nombre es un TCPIPService

Contiene:

1. Un elemento <name>, que contiene el nombre de recurso
2. Un elemento <transport_handler_list> opcional. Cada <transport_handler_list> contiene uno o más elementos <handler>.
Si no codifica un elemento <transport_handler_list>, los únicos manejadores de mensajes que se invocan cuando se utiliza el transporte con nombre son los que están especificados en el elemento <service>.

Ejemplo

```
<named_transport_entry type="http">
  <name>PORT80</name>
  <transport_handler_list>
    <handler><program>HANDLER1</program><handler_parameter_list/></handler>
    <handler><program>HANDLER2</program><handler_parameter_list/></handler>
  </transport_handler_list>
</named_transport_entry>
```

En este ejemplo, los manejadores de mensajes especificados (HANDLER1 y HANDLER2) se invocan para mensajes recibidos en TCPIPService con el nombre PORT80.

Elemento <provider_pipeline>:

Especifica el elemento raíz del documento XML que describe la configuración de la interconexión de CICS para un proveedor de servicios web.

Se usa en lo siguiente:

- Proveedor de servicios

Contiene:

1. Elemento <cics_mtom_handler> opcional
2. Elemento <transport> opcional
3. Elemento <service>

4. Elemento <apphandler> opcional
5. Elemento <apphandler_class> opcional
6. Elemento <service_parameter_list> opcional, que contiene elementos XML que pasan a estar disponibles a todos los manejadores de mensajes en la interconexión en el contenedor DFH-SERVICEPLIST.

Ejemplo

```
<provider_pipeline>
  <service>
    ...
  </service>
  <apphandler>DFHPITP</apphandler>
</provider_pipeline>
```

Elemento <provider_pipeline_json>:

Especifica el elemento raíz del documento XML que describe la configuración de la interconexión de CICS para un proveedor de servicios web JSON z/OS Connect.

Este difiere del elemento <provider_pipeline> en que los programas de manejador no se pueden definir. Este estilo de interconexión se utiliza como contenedor para los recursos **WEBSERVICE** que utiliza z/OS Connect. Un intento de iniciar una interconexión <provider_pipeline_json> utilizando cualquier otra cosa que no sea z/OS Connect dará como resultado un error. El recurso **PIPELINE** resultante no se puede utilizar como destino de un recurso **USAGE(PIPELINE) URIMAP**. Sólo se puede utilizar con recursos **USAGE(JVMSEVER) URIMAP**.

Se usa en lo siguiente:

- Proveedor de servicios

Atributos:

java_parser=**{yes|no}**

Seleccione el tipo de analizador JSON que se utiliza para procesar mensajes de entrada.

Los valores posibles son:

yes Realice el análisis de JSON utilizando Java en el servidor JVM. Es el valor predeterminado.

no Realice un análisis que no sea de Java del mensaje JSON.

Nota: El atributo java_parser es opcional. Si no lo proporciona, el comportamiento predeterminado es analizar el mensaje JSON utilizando Java, en el servidor JVM. Esto sería lo mismo que especificar java_parser="yes".

java_generator=**{sí|no}**

Seleccione el tipo de generador JSON que se utiliza para generar mensajes de salida.

Los valores posibles son:

yes Lleve a cabo la generación JSON utilizando Java dentro del servidor JVM.

no Lleve a cabo la generación no Java de los mensajes JSON. Es el valor predeterminado.

Contiene:

- Un elemento <jvmserver>, que contiene el nombre del recurso JVMSERVER en el que está configurado z/OS Connect.

Ejemplo que utiliza el análisis de Java

```
<provider_pipeline_json java_parser="yes">
  <jvmserver>DFHWLP</jvmserver>
</provider_pipeline_json>
```

Ejemplo que utiliza el análisis que no es de Java

```
<provider_pipeline_json java_parser="no">
  <jvmserver>DFHWLP</jvmserver>
</provider_pipeline_json>
```

Elemento <terminal_handler>:

Contiene la definición del manejador de mensajes de terminal de la interconexión del proveedor de servicios.

Se usa en lo siguiente:

- Proveedor de servicios

Contenido en:

- Elemento <service>

Contiene:

Uno de los siguientes elementos:

```
<handler>
<cics_json_handler_java>
<cics_soap_1.1_handler>
<cics_soap_1.2_handler>
<cics_soap_1.1_handler_java>
<cics_soap_1.2_handler_java>
```

Si espera que la interconexión procese los mensajes SOAP 1.1 y SOAP 1.2, debe utilizar el elemento <cics_soap_1.2_handler> o <cics_soap_1.2_handler_java>.

Recuerde: En un proveedor de servicios, puede especificar <cics_soap_1.1_handler> y <cics_soap_1.2_handler> en el elemento <service_handler_list>, así como en el elemento <terminal_handler>. Sin embargo, en un proveedor de servicios, sólo puede especificar <cics_soap_1.1_handler_java> y <cics_soap_1.2_handler_java> en el elemento <terminal_handler>.

Ejemplo

```
<terminal_handler>
  <cics_soap_1.1_handler>
  ...
  </cics_soap_1.1_handler>
</service_handler_list>
```

Ejemplo: habilitación de un proceso no Java de mensajes JSON

Para habilitar un proceso no Java de mensajes JSON, especifique el programa de controlador de terminal como DFHPIJT:

```
<terminal_handler>
  <handler>
    <program>DFHPIJT</program><handler_parameter_list/>
  </handler>
</terminal_handler>
```

Nota: Cuando utiliza DFHPIJT como el controlador de terminal, no define un manejador de aplicación en el archivo de configuración de interconexión, es decir, el archivo de configuración de interconexión no debe contener un elemento <apphandler>. Si se especifica un manejador de aplicación, no se invoca.

Elemento <transport_handler_list>:

Contiene una lista de manejadores de mensajes que se invocan cuando se utiliza un recurso con nombre.

- Para el transporte MQ, el recurso con nombre es el nombre de la cola de entrada local.
- Para el transporte HTTP, el recurso es el TCPIPService que define el puerto en el que se ha recibido la solicitud.

Se usa en lo siguiente:

- Proveedor de servicios

Contenido en:

- Elemento <named_transport_entry>

Contiene:

- Uno o más elementos <handler>.

Ejemplo

```
<transport_handler_list>
  <handler>
    ...
  </handler>
  <handler>
    ...
  </handler>
</transport_handler_list>
```

Elementos utilizados en solicitantes de servicio

Algunos de los elementos XML utilizados en un archivo de configuración de interconexión se aplican únicamente a interconexiones de solicitante de servicio.

Elemento <requester_pipeline>:

Elemento raíz del documento XML que describe la configuración de una interconexión en un solicitante de servicio.

Se usa en lo siguiente:

- Solicitante de servicios

Contiene:

1. Elemento <service> opcional
2. Elemento <transport> opcional
3. Elemento <cics_mtom_handler> opcional
4. Elemento <service_parameter_list> opcional, que contiene elementos XML disponibles a los manejadores de mensajes en el contenedor DFH-SERVICEPLIST.

Ejemplo

```
<requester_pipeline>
  <service>
    <service_handler_list>
      <cics_soap_1.1_handler/>
    </service_handler_list>
  </service>
</requester_pipeline>
```

Elementos utilizados en las interconexiones de solicitante de servicio y proveedor de servicios

Algunos de los elementos XML utilizados en un archivo de configuración de interconexión se aplican a las interconexiones de solicitante de servicio y proveedor de servicios.

Elemento <addressing>:

Especifica el soporte para Web Services Addressing en un proceso SOAP basado en Java.

Se usa en lo siguiente:

- Proveedor de servicios
- Solicitante de servicio

Contenido en:

Elemento <cics_soap_1.1_handler_java>

Elemento <cics_soap_1.2_handler_java>

Contiene:

Un elemento <namespace>. En un proveedor de servicios este elemento es opcional. El elemento contiene uno de los dos esquemas WS-Addressing soportados por CICS. Para los mensajes de entrada, Axis2 soporta ambas especificaciones. Para mensajes de salida, se utiliza el espacio de nombres especificado en este elemento. Si no especifica este elemento o tiene dos elementos, CICS utiliza la misma especificación en el mensajes de salida que en el mensaje de entrada. En un solicitante de servicio, este elemento es necesario y puede especificar únicamente un espacio de nombres para el mensaje de salida.

Este ejemplo muestra la configuración para una interconexión de proveedor de servicios, donde se soportan ambas especificaciones de WS-Addressing. CICS utiliza la misma especificación en el mensajes de salida que en el mensaje de entrada. Puede obtener los mismos resultados especificando un elemento <addressing> vacío.

```

<addressing>
  <namespace>http://www.w3.org/2005/08/addressing</namespace>
  <namespace>http://schemas.xmlsoap.org/ws/2004/08/addressing</namespace>
</addressing>

```

Elemento <cics_json_handler_java>:

Especifica los atributos del programa del controlador para mensajes JSON en interconexiones JSON basadas en Java.

Se usa en lo siguiente:

- Proveedor de servicios

Contenido en:

The <service_handler_list> element

The <terminal_handler> element

Contiene:

1. Un elemento <jvmserver>.
2. Un elemento <repository> opcional.

Ejemplo

El siguiente ejemplo muestra el XML del controlador JSON basado en Java y sus elementos anidados:

```

<?xml version="1.0" encoding="EBCDIC-CP-US"?>
<provider_pipeline
  xmlns="http://www.ibm.com/software/htp/cics/pipeline">
  <service>
    <terminal_handler>
      <cics_json_handler_java>
        <jvmserver>DFHAXIS</jvmserver>
        <repository>/usr/lpp/cicsts/cicsts54/lib/pipeline/repository</repository>
      </cics_json_handler_java>
    </terminal_handler>
  </service>
  <apphandler_class>com.ibm.cicsts.axis2.CICSAxis2ApplicationHandler</apphandler_class>
</provider_pipeline>

```

Elemento <cics_soap_1.1_handler>:

Especifica los atributos del programa del controlador para mensajes SOAP 1.1 en interconexiones que no son Java.

Se usa en lo siguiente:

- Solicitante de servicio
- Proveedor de servicios

Contenido en:

Elemento <service_handler_list>

Elemento <terminal_handler>

Contiene:

Cero, uno o más elementos <headerprogram>. Cada <headerprogram> contiene:

1. Un elemento <program_name>, que contiene el nombre de un programa de proceso de cabeceras

2. Un elemento `<namespace>`, que se utiliza con el siguiente elemento `<localname>` para determinar qué bloques de cabecera en un mensaje SOAP debe procesar el programa de proceso de cabeceras. El elemento `<namespace>` contiene el URI (Identificador uniforme de recursos) del espacio de nombres del bloque de cabecera.
3. Un elemento `<localname>`, que se utiliza con el elemento `<namespace>` precedente para determinar qué bloques de cabecera en un mensaje SOAP debe procesar el programa de proceso de cabeceras. El `<localname>` contiene el nombre de elemento del bloque de cabecera.

Por ejemplo, tenga en cuenta este bloque de cabecera:

```
<t:myheaderblock xmlns:t="http://mynamespace" ...> .... </t:myheaderblock>
```

- El nombre del espacio de nombres es `http://mynamespace`
- El nombre de elemento es `myheaderblock`

Para hacer que el programa de cabecera coincida con este bloque de cabecera, codifique los elementos `<namespace>` y `<localname>` como éste:

```
<namespace>http://mynamespace</namespace>
<localname>myheaderblock</localname>
```

Puede codificar un asterisco (*) en el elemento `<localname>` para indicar que deben procesarse todos los bloques de cabeceras en el espacio de nombres cuyos nombres comienzan con una serie de caracteres dados. Por ejemplo:

```
<namespace>http://mynamespace</namespace>
<localname>myhead*</localname>
```

Cuando utiliza el asterisco en el elemento `<localname>`, una cabecera en un mensaje puede coincidir con más de un elemento a `<headerprogram>`. Por ejemplo, este bloque de cabecera

```
<t:myheaderblock xmlns:t="http://mynamespace" ...> .... </myheaderblock>
```

coincide con todos los elementos `<headerprogram>` siguientes:

```
<headerprogram>
  <program_name>HDRPROG1</program_name>
  <namespace>http://mynamespace</namespace>
  <localname>*</localname>
  <mandatory>>false</mandatory>
</headerprogram>
<headerprogram>
  <program_name>HDRPROG2</program_name>
  <namespace>http://mynamespace</namespace>
  <localname>myhead*</localname>
  <mandatory>>false</mandatory>
</headerprogram>
<headerprogram>
  <program_name>HDRPROG3</program_name>
  <namespace>http://mynamespace</namespace>
  <localname>myheaderblock</localname>
  <mandatory>>false</mandatory>
</headerprogram>
```

Cuando éste es el caso, el programa de cabecera que se ejecuta es el especificado en el elemento `<headerprogram>` en el que el nombre de elemento del bloque de cabecera se indica de forma más precisa. En el ejemplo, es HDRPROG3.

Cuando el mensaje SOAP contiene más de una cabecera, el programa de proceso de cabeceras se invoca una vez para cada cabecera coincidente, pero la secuencia en la que se procesan las cabeceras no está definida.

Si codifica dos o más elementos <headerprogram> que contienen los mismos elementos <namespace> y <localname>, pero que especifican programas de cabecera diferentes, sólo se ejecutará uno de los programas de cabecera, pero no está definido cuál de los programas se ejecutará.

4. Un elemento <mandatory>, que contiene un valor booleano XML (true o false). De lo contrario, puede codificar los valores como 1 o 0 respectivamente.

true

Durante el proceso de solicitud de servicio en una interconexión de proveedor de servicios, y el proceso de respuesta de servicio en una interconexión de solicitante de servicio, el programa de proceso de cabeceras se invocará al menos una vez, aunque ninguna de las cabeceras en los mensajes SOAP coincida con los elementos <namespace> y <localname>:

- Si no coincide ninguna de las cabeceras, el programa de proceso de cabeceras se invoca una vez.
- Si alguna de las cabeceras coincide, el programa de proceso de cabeceras se invoca una vez para cada cabecera coincidente.

Durante el proceso de solicitud de servicio en una interconexión de solicitante de servicio, el programa de proceso de cabeceras se invocará al menos una vez, aunque el mensaje SOAP que crea CICS no tiene cabeceras inicialmente. Si desea añadir cabeceras al mensaje, debe asegurarse de que se invoca al menos un programa de proceso de cabeceras, especificando <mandatory>true</mandatory> o <mandatory>1</mandatory>.

false

El programa de proceso de cabeceras se invocará sólo si una o varias cabeceras de los mensajes SOAP coinciden con los elementos <namespace> y <localname>:

- Si no coincide ninguna de las cabeceras, el programa de proceso de cabeceras no se invoca.
- Si alguna de las cabeceras coincide, el programa de proceso de cabeceras se invoca una vez para cada cabecera coincidente.

Ejemplo

```
<cics_soap_1.1_handler>
  <headerprogram>
    <program_name> ... </program_name>
    <namespace>...</namespace>
    <localname>...</localname>
    <mandatory>true</mandatory>
  </headerprogram>
</cics_soap_1.1_handler>
```

Elemento <cics_soap_1.1_handler_java>:

Especifica los atributos del programa del controlador para mensajes SOAP en interconexiones SOAP basadas en Java.

Se usa en lo siguiente:

- Solicitante de servicio
- Proveedor de servicios

Contenido en:

Elemento <service_handler_list>

Elemento <terminal_handler>

Contiene:

1. Un elemento <jvmserver>.
2. Un elemento <repository> opcional.
3. Un elemento <addressing> opcional. Si habilita Web Services Addressing en Axis2, no utilice el programa de proceso de cabeceras DFHWSADH.
4. Cero, uno o más elementos <headerprogram>. Cada elemento <headerprogram> contiene:
 - a. Un elemento <program_name>, que contiene el nombre de un programa de proceso de cabeceras. Puede escribir manejadores Axis2 en Java para procesar cabeceras SOAP.
 - b. Un elemento <namespace>, que se utiliza con el siguiente elemento <localname> para determinar qué bloques de cabecera en un mensaje SOAP debe procesar el programa de proceso de cabeceras. El elemento <namespace> contiene el URI (Identificador uniforme de recursos) del espacio de nombres del bloque de cabecera.
 - c. Un elemento <localname>, que se utiliza con el elemento <namespace> precedente para determinar qué bloques de cabecera en un mensaje SOAP debe procesar el programa de proceso de cabeceras. El <localname> contiene el nombre de elemento del bloque de cabecera.

Por ejemplo, tenga en cuenta este bloque de cabecera:

```
<t:myheaderblock xmlns:t="http://mynamespace" ...> .... </t:myheaderblock>
```

El nombre del espacio de nombres es http://mynamespace y el nombre de elemento es myheaderblock.

Para hacer que el programa de cabecera coincida con este bloque de cabecera, codifique los elementos <namespace> y <localname> como éste:

```
<namespace>http://mynamespace</namespace>
<localname>myheaderblock</localname>
```

Puede codificar un asterisco (*) en el elemento <localname> para indicar que deben procesarse todos los bloques de cabeceras en el espacio de nombres cuyos nombres comienzan con una serie de caracteres dados. Por ejemplo:

```
<namespace>http://mynamespace</namespace>
<localname>myhead*</localname>
```

Cuando utiliza el asterisco en el elemento <localname>, una cabecera en un mensaje puede coincidir con más de un elemento a <headerprogram>. Por ejemplo, este bloque de cabecera:

```
<t:myheaderblock xmlns:t="http://mynamespace" ...> .... </myheaderblock>
```

coincide con todos los elementos <headerprogram> siguientes:

```
<headerprogram>
  <program_name>HDRPROG1</program_name>
  <namespace>http://mynamespace</namespace>
  <localname>*</localname>
  <mandatory>>false</mandatory>
</headerprogram>
<headerprogram>
  <program_name>HDRPROG2</program_name>
  <namespace>http://mynamespace</namespace>
  <localname>myhead*</localname>
```

```

    <mandatory>false</mandatory>
  </headerprogram>
</headerprogram>
  <program_name>HDRPROG3</program_name>
  <namespace>http://mynamespace</namespace>
  <localname>myheaderblock</localname>
  <mandatory>false</mandatory>
</headerprogram>

```

Cuando éste es el caso, el programa de cabecera que se ejecuta es el especificado en el elemento `<headerprogram>` en el que el nombre de elemento del bloque de cabecera se indica de forma más precisa. En el ejemplo, es HDRPROG3.

Cuando el mensaje SOAP contiene más de una cabecera, el programa de proceso de cabeceras se invoca una vez para cada cabecera coincidente, pero la secuencia en la que se procesan las cabeceras no está definida.

Si codifica dos o más elementos `<headerprogram>` que contienen los mismos elementos `<namespace>` y `<localname>`, pero que especifican programas de cabecera diferentes, sólo se ejecutará uno de los programas de cabecera, pero no está definido cuál de los programas se ejecutará.

- d. Un elemento `<mandatory>`, que contiene un valor booleano XML (`true` o `false`). De lo contrario, puede codificar los valores como 1 o 0 respectivamente.

true

Durante el proceso de solicitud de servicio en una interconexión de proveedor de servicios, y el proceso de respuesta de servicio en una interconexión de solicitante de servicio, el programa de proceso de cabeceras se invocará al menos una vez, aunque ninguna de las cabeceras en los mensajes SOAP coincida con los elementos `<namespace>` y `<localname>`:

- Si no coincide ninguna de las cabeceras, el programa de proceso de cabeceras se invoca una vez.
- Si alguna de las cabeceras coincide, el programa de proceso de cabeceras se invoca una vez para cada cabecera coincidente.

Durante el proceso de solicitud de servicio en una interconexión de solicitante de servicio, el programa de proceso de cabeceras se invocará al menos una vez, aunque el mensaje SOAP que crea CICS no tiene cabeceras inicialmente. Si desea añadir cabeceras al mensaje, debe asegurarse de que se invoca al menos un programa de proceso de cabeceras, especificando `<mandatory>true</mandatory>` o `<mandatory>1</mandatory>`.

false

El programa de proceso de cabeceras se invocará sólo si una o varias cabeceras de los mensajes SOAP coinciden con los elementos `<namespace>` y `<localname>`:

- Si no coincide ninguna de las cabeceras, el programa de proceso de cabeceras no se invoca.
- Si alguna de las cabeceras coincide, el programa de proceso de cabeceras se invoca una vez para cada cabecera coincidente.

Ejemplo

El siguiente ejemplo muestra el XML para el controlador SOAP basado en Java y sus elementos anidados:


```

<cics_soap_1.1_handler_java>
  <jvmserver>JVMSESV1</jvmserver>
  <headerprogram>
    <program_name>HDRPROG4</program_name>
    <namespace>http://mynamespace</namespace>
    <localname>myheaderblock</localname>
    <mandatory>true</mandatory>
  </headerprogram>
</cics_soap_1.1_handler_java>

```

Elemento **<cics_soap_1.2_handler>**:

Especifica los atributos del programa del controlador para mensajes SOAP 1.2 en interconexiones que no son Java.

Se usa en lo siguiente:

- Solicitante de servicio
- Proveedor de servicios

Contenido en:

Elemento **<service_handler_list>**

Elemento **<terminal_handler>**

Contiene:

Cero, uno o más elementos **<headerprogram>**. Cada **<headerprogram>** contiene:

1. Un elemento **<program_name>**, que contiene el nombre de un programa de proceso de cabeceras
2. Un elemento **<namespace>**, que se utiliza con el siguiente elemento **<localname>** para determinar qué bloques de cabecera en un mensaje SOAP debe procesar el programa de proceso de cabeceras. El elemento **<namespace>** contiene el URI (Identificador uniforme de recursos) del espacio de nombres del bloque de cabecera.
3. Un elemento **<localname>**, que se utiliza con el elemento **<namespace>** precedente para determinar qué bloques de cabecera en un mensaje SOAP debe procesar el programa de proceso de cabeceras. El **<localname>** contiene el nombre de elemento del bloque de cabecera.

Por ejemplo, tenga en cuenta este bloque de cabecera:

```
<t:myheaderblock xmlns:t="http://mynamespace" ...> .... </t:myheaderblock>
```

- El nombre del espacio de nombres es **http://mynamespace**
- El nombre de elemento es **myheaderblock**

Para hacer que el programa de cabecera coincida con este bloque de cabecera, codifique los elementos **<namespace>** y **<localname>** como éste:

```

<namespace>http://mynamespace</namespace>
<localname>myheaderblock</localname>

```

Puede codificar un asterisco (*) en el elemento **<localname>** para indicar que deben procesarse todos los bloques de cabeceras en el espacio de nombres cuyos nombres comienzan con una serie de caracteres dados. Por ejemplo:

```

<namespace>http://mynamespace</namespace>
<localname>myhead*</localname>

```

Cuando utiliza el asterisco en el elemento **<localname>**, una cabecera en un mensaje puede coincidir con más de un elemento a **<headerprogram>**. Por ejemplo, este bloque de cabecera

```
<t:myheaderblock xmlns:t="http://mynamespace" ...> .... </myheaderblock>
```

coincide con todos los elementos <headerprogram> siguientes:

```
<headerprogram>
  <program_name>HDRPROG1</program_name>
  <namespace>http://mynamespace</namespace>
  <localname>*</localname>
  <mandatory>>false</mandatory>
</headerprogram>
<headerprogram>
  <program_name>HDRPROG2</program_name>
  <namespace>http://mynamespace</namespace>
  <localname>myhead*</localname>
  <mandatory>>false</mandatory>
</headerprogram>
<headerprogram>
  <program_name>HDRPROG3</program_name>
  <namespace>http://mynamespace</namespace>
  <localname>myheaderblock</localname>
  <mandatory>>false</mandatory>
</headerprogram>
```

Cuando éste es el caso, el programa de cabecera que se ejecuta es el especificado en el elemento <headerprogram> en el que el nombre de elemento del bloque de cabecera se indica de forma más precisa. En el ejemplo, es HDRPROG3.

Cuando el mensaje SOAP contiene más de una cabecera, el programa de proceso de cabeceras se invoca una vez para cada cabecera coincidente, pero la secuencia en la que se procesan las cabeceras no está definida.

Si codifica dos o más elementos <headerprogram> que contienen los mismos elementos <namespace> y <localname>, pero que especifican programas de cabecera diferentes, sólo se ejecutará uno de los programas de cabecera, pero no está definido cuál de los programas se ejecutará.

4. Un elemento <mandatory>, que contiene un valor booleano XML (true o false). De lo contrario, puede codificar los valores como 1 o 0 respectivamente.

true

Durante el proceso de solicitud de servicio en una interconexión de proveedor de servicios, y el proceso de respuesta de servicio en una interconexión de solicitante de servicio, el programa de proceso de cabeceras se invocará al menos una vez, aunque ninguna de las cabeceras en los mensajes SOAP coincida con los elementos <namespace> y <localname>:

- Si no coincide ninguna de las cabeceras, el programa de proceso de cabeceras se invoca una vez.
- Si alguna de las cabeceras coincide, el programa de proceso de cabeceras se invoca una vez para cada cabecera coincidente.

Durante el proceso de solicitud de servicio en una interconexión de solicitante de servicio, el programa de proceso de cabeceras se invocará al menos una vez, aunque el mensaje SOAP que crea CICS no tiene cabeceras inicialmente. Si desea añadir cabeceras al mensaje, debe asegurarse de que se invoca al menos un programa de proceso de cabeceras, especificando <mandatory>true</mandatory> o <mandatory>1</mandatory>.

false

El programa de proceso de cabeceras se invocará sólo si una o varias cabeceras de los mensajes SOAP coinciden con los elementos <namespace> y <localname>:

- Si no coincide ninguna de las cabeceras, el programa de proceso de cabeceras no se invoca.
- Si alguna de las cabeceras coincide, el programa de proceso de cabeceras se invoca una vez para cada cabecera coincidente.

Ejemplo

```
<cics_soap_1.2_handler>
  <headerprogram>
    <program_name> ... </program_name>
    <namespace>...</namespace>
    <localname>...</localname>
    <mandatory>true</mandatory>
  </headerprogram>
</cics_soap_1.2_handler>
```

Elemento **<cics_soap_1.2_handler_java>**:

Especifica los atributos del programa del controlador para mensajes SOAP 1.2 en interconexiones SOAP basadas en Java.

Se usa en lo siguiente:

- Solicitante de servicio
- Proveedor de servicios

Contenido en:

Elemento **<service_handler_list>**

Elemento **<terminal_handler>**

Contiene:

1. Un elemento **<jvmserver>**.
2. Un elemento **<repository>** opcional.
3. Un elemento **<addressing>** opcional. Si habilita el soporte para Web Services Addressing en Axis2, no utilice el programas de proceso de cabeceras. Puede escribir manejadores Axis2 en Java para procesar cabeceras SOAP.
4. Cero, uno o más elementos **<headerprogram>**. Cada elemento **<headerprogram>** contiene:
 - a. Un elemento **<program_name>**, que contiene el nombre de un programa de proceso de cabeceras
 - b. Un elemento **<namespace>**, que se utiliza con el siguiente elemento **<localname>** para determinar qué bloques de cabecera en un mensaje SOAP debe procesar el programa de proceso de cabeceras. El elemento **<namespace>** contiene el URI (Identificador uniforme de recursos) del espacio de nombres del bloque de cabecera.
 - c. Un elemento **<localname>**, que se utiliza con el elemento **<namespace>** precedente para determinar qué bloques de cabecera en un mensaje SOAP debe procesar el programa de proceso de cabeceras. El **<localname>** contiene el nombre de elemento del bloque de cabecera.

Por ejemplo, tenga en cuenta este bloque de cabecera:

```
<t:myheaderblock xmlns:t="http://mynamespace" ...> .... </t:myheaderblock>
```

El nombre del espacio de nombres es `http://mynamespace` y el nombre de elemento es `myheaderblock`.

Para hacer que el programa de cabecera coincida con este bloque de cabecera, codifique los elementos <namespace> y <localname> como éste:

```
<namespace>http://mynamespace</namespace>
<localname>myheaderblock</localname>
```

Puede codificar un asterisco (*) en el elemento <localname> para indicar que deben procesarse todos los bloques de cabeceras en el espacio de nombres cuyos nombres comienzan con una serie de caracteres dados. Por ejemplo:

```
<namespace>http://mynamespace</namespace>
<localname>myhead*</localname>
```

Cuando utiliza el asterisco en el elemento <localname>, una cabecera en un mensaje puede coincidir con más de un elemento a <headerprogram>. Por ejemplo, este bloque de cabecera:

```
<t:myheaderblock xmlns:t="http://mynamespace" ...> .... </myheaderblock>
```

coincide con todos los elementos <headerprogram> siguientes:

```
<headerprogram>
  <program_name>HDRPROG1</program_name>
  <namespace>http://mynamespace</namespace>
  <localname>*</localname>
  <mandatory>false</mandatory>
</headerprogram>
<headerprogram>
  <program_name>HDRPROG2</program_name>
  <namespace>http://mynamespace</namespace>
  <localname>myhead*</localname>
  <mandatory>false</mandatory>
</headerprogram>
<headerprogram>
  <program_name>HDRPROG3</program_name>
  <namespace>http://mynamespace</namespace>
  <localname>myheaderblock</localname>
  <mandatory>false</mandatory>
</headerprogram>
```

Cuando éste es el caso, el programa de cabecera que se ejecuta es el especificado en el elemento <headerprogram> en el que el nombre de elemento del bloque de cabecera se indica de forma más precisa. En el ejemplo, es HDRPROG3.

Cuando el mensaje SOAP contiene más de una cabecera, el programa de proceso de cabeceras se invoca una vez para cada cabecera coincidente, pero la secuencia en la que se procesan las cabeceras no está definida.

Si codifica dos o más elementos <headerprogram> que contienen los mismos elementos <namespace> y <localname>, pero que especifican programas de cabecera diferentes, sólo se ejecutará uno de los programas de cabecera, pero no está definido cuál de los programas se ejecutará.

- d. Un elemento <mandatory>, que contiene un valor booleano XML (true o false). De lo contrario, puede codificar los valores como 1 o 0 respectivamente.

true

Durante el proceso de solicitud de servicio en una interconexión de proveedor de servicios, y el proceso de respuesta de servicio en una interconexión de solicitante de servicio, el programa de proceso de cabeceras se invocará al menos una vez, aunque ninguna de las cabeceras en los mensajes SOAP coincida con los elementos <namespace> y <localname>:

- Si no coincide ninguna de las cabeceras, el programa de proceso de cabeceras se invoca una vez.
- Si alguna de las cabeceras coincide, el programa de proceso de cabeceras se invoca una vez para cada cabecera coincidente.

Durante el proceso de solicitud de servicio en una interconexión de solicitante de servicio, el programa de proceso de cabeceras se invocará al menos una vez, aunque el mensaje SOAP que crea CICS no tiene cabeceras inicialmente. Si desea añadir cabeceras al mensaje, debe asegurarse de que se invoca al menos un programa de proceso de cabeceras, especificando `<mandatory>true</mandatory>` o `<mandatory>1</mandatory>`.

false

El programa de proceso de cabeceras se invocará sólo si una o varias cabeceras de los mensajes SOAP coinciden con los elementos `<namespace>` y `<localname>`:

- Si no coincide ninguna de las cabeceras, el programa de proceso de cabeceras no se invoca.
- Si alguna de las cabeceras coincide, el programa de proceso de cabeceras se invoca una vez para cada cabecera coincidente.

Ejemplo

El siguiente ejemplo muestra el XML para el controlador SOAP basado en Java y sus elementos anidados:

```
<cics_soap_1.2_handler_java>
  <jvmserver>JVMSESV1</jvmserver>
  <headerprogram>
    <program_name>HDRPROG4</program_name>
    <namespace>http://mynamespace</namespace>
    <localname>myheaderblock</localname>
    <mandatory>true</mandatory>
  </headerprogram>
</cics_soap_1.2_handler_java>
```

Elemento `<default_http_transport_handler_list>`:

Especifica los manejadores de mensajes que se invocan de forma predeterminada cuando el transporte HTTP está en uso.

En un proveedor de servicios, los manejadores de mensajes especificados en esta lista se invoca únicamente si la lista de manejadores definidos en un elemento `<named_transport_entry>` es menos específica.

Se usa en lo siguiente:

- Proveedor de servicios
- Solicitante de servicio

Contenido en:

- Elemento `<transport>`

Contiene:

- Uno o más elementos `<handler>`.

Ejemplo

```
<default_http_transport_handler_list>
  <handler>
    ...
  </handler>
  <handler>
    ...
  </handler>
</default_http_transport_handler_list>
```

Elemento **<default_mq_transport_handler_list>**:

Especifica los manejadores de mensajes que se invocan de forma predeterminada cuando el transporte de WebSphere MQ está en uso.

En un proveedor de servicios, los manejadores de mensajes especificados en esta lista se invoca únicamente si la lista de manejadores definidos en un elemento **<named_transport_entry>** es menos específica.

Se usa en lo siguiente:

- Proveedor de servicios
- Solicitante de servicio

Contenido en:

- Elemento **<transport>**

Contiene:

- Uno o más elementos **<handler>**.

Ejemplo

```
<default_mq_transport_handler_list>
  <handler>
    ...
  </handler>
  <handler>
    ...
  </handler>
</default_mq_transport_handler_list>
```

Elemento **<default_transport_handler_list>**:

Especifica los manejadores de mensajes que se invocan de forma predeterminada cuando algún transporte está en uso.

En un proveedor de servicios, los manejadores de mensajes especificados en esta lista se invocan cuando la lista de manejadores definida en cualquiera de los siguientes elementos es menos específica:

```
  <default_http_transport_handler_list>
  <default_mq_transport_handler_list>
  <named_transport_entry>
```

Se usa en lo siguiente:

- Proveedor de servicios
- Solicitante de servicio

Contenido en:

- Elemento <transport>

Contiene:

- Uno o más elementos <handler>.

Ejemplo

```
<default_transport_handler_list>
  <handler>
    <program>HANDLER1</program>
    <handler_parameter_list/>
  </handler>
  <handler>
    <program>HANDLER2</program>
    <handler_parameter_list/>
  </handler>
</default_transport_handler_list>
```

Elemento <handler>:

Especifica los atributos de un programa manejador de mensajes.

Algunos programas de manejador proporcionados por CICS no utilizan el elemento <handler>. Por ejemplo, los programas de manejador de mensajes proporcionados por CICS se definen utilizando los elementos <cics_soap_1.1_handler>, <cics_soap_1.2_handler>, <cics_soap_1.1_handler_java> y <cics_soap_1.2_handler_java>.

Se usa en lo siguiente:

- Proveedor de servicios
- Solicitante de servicio

Contenido en:

```
<default_transport_handler_list>
  <transport_handler_list>
  <service_handler_list>
  <terminal_handler>
  <default_http_transport_handler_list>
  <default_mq_transport_handler_list>
```

Contiene:

1. Un elemento <program>, que contiene el nombre del programa de manejador
2. Un elemento <handler_parameter_list>, que contiene elementos XML que están disponibles para los manejadores de mensajes en el contenedor DFH-HANDLERPLIST.

Ejemplo

```
<?xml version="1.0"?>
<provider_pipeline>
  xmlns="http://www.ibm.com/software/http/cics/pipeline">
  <service>
    <service_handler_list>
      <handler>
        <program>MYPROG</program>
        <handler_parameter_list><output print="yes"/></handler_parameter_list>
      </handler>
```

```

</service_handler_list>
<terminal_handler>
  <cics_soap_1.1_handler>
    ...
  </cics_soap_1.1_handler>
</terminal_handler>
</service
  <apphandler>DFHPITP</apphandler>
</provider_pipeline>

```

En este ejemplo, el programa de manejador es MYPROG. La lista de parámetros de manejador consta de un único elemento <output>; MYPROG conoce el contenido de la lista de parámetros de manejador.

Elemento <jvmserver>:

Especifica el nombre del recurso JVMSERVER.

Este elemento identifica el nombre del recurso JVMSERVER, que procesará la solicitud. Si no se suministra un valor, se generará un mensaje de error y PIPELINE se instala con el estado DISABLED.

Se usa en lo siguiente:

- Proveedor de servicios
- Solicitante de servicio

Contenido en:

- Elemento de The <cics_json_handler_java> element
- Elemento de The <cics_soap_1.1_handler_java> element
- Elemento de The <cics_soap_1.2_handler_java> element
- Elemento de

Ejemplo

```
<jvmserver>JVMSERVER_NAME</jvmserver>
```

El elemento <repository>:

Especifica el nombre de directorio del repositorio Axis2.

Este elemento opcional especifica el nombre de directorio del repositorio Axis2. Si utiliza esta opción, debe especificar The <jvmserver> element de antemano en el XML del manejador. Si no se proporciona el elemento, se utilizará el repositorio de ejemplo. Cuando se instala CICS Transaction Server, el repositorio de Axis2 de ejemplo se instala en el directorio /usr/lpp/cicsts/cicsts54/lib/pipeline/repository, donde /usr/lpp/cicsts/cicsts54 es el directorio de instalación predeterminado de archivos de CICS en z/OS UNIX.

Se usa en lo siguiente:

- Proveedor de servicios
- Solicitante de servicios

Contenido en:

- The <cics_json_handler_java> element
- The <cics_soap_1.1_handler_java> element
- The <cics_soap_1.2_handler_java> element

Ejemplo

```
<cics_soap_1.1_handler_java>  
  <jvmserver>JVMSERV1</jvmserver>  
  <repository>/lib/pipeline/repository</repository>  
</cics_soap_1.1_handler_java>
```

El elemento **<service>**:

Especifica los manejadores de mensajes que se invocan para cada solicitud.

Se usa en lo siguiente:

- Proveedor de servicios
- Solicitante de servicios

Contenido en:

```
  <provider_pipeline>  
  <requester_pipeline>
```

Contiene:

1. Elemento **<service_handler_list>**
2. En proveedor de servicios sólo, un elemento **<terminal_handler>**

Ejemplo

```
<service>  
  <service_handler_list>  
    ...  
  </service_handler_list>  
  <terminal_handler>  
    ...  
  </terminal_handler>  
</service>
```

Elemento **<service_handler_list>**:

Especifica una lista de manejadores de mensajes invocado para cada solicitud.

Se usa en lo siguiente:

- Proveedor de servicios
- Solicitante de servicio

Contenido en:

- Elemento **<service>**

Contiene:

Uno o varios de los siguientes elementos:

```
  <cics_soap_1.1_handler>  
  <cics_soap_1.2_handler>  
  <cics_soap_1.1_handler_java>  
  <cics_soap_1.2_handler_java>  
  <handler>  
  <wsse_handler>
```

Determine el orden en el que se llama a cada manejador en el tiempo de ejecución por el orden que especifica en los elementos de manejador en el elemento `<service_handler_list>`. Por ejemplo, si la interconexión soporta WS-Security, los mensajes SOAP cifrados permanecen cifrados hasta que se llama al elemento `<wsse_handler>`. Por lo tanto, debe especificar el elemento `<wsse_handler>` antes que cualquier otro programa de manejador que procese mensajes no cifrados.

El elemento `<service_handler_list>` para un proveedor de servicios no puede contener los elementos `<cics_soap_1.1_handler_java>` y `<cics_soap_1.2_handler_java>`, porque estos elementos deben especificarse en el elemento `<terminal_handler>` para interconexiones basadas en Java. Un solicitante de servicio puede contener `<cics_soap_1.1_handler_java>` y `<cics_soap_1.2_handler_java>`, sin embargo, si se utilizan estos elementos, deben ser el primer elemento listado en el elemento `<service_handler_list>`.

Si espera que la interconexión procese los mensajes SOAP 1.1 y SOAP 1.2, debe utilizar el elemento `<cics_soap_1.2_handler>` o `<cics_soap_1.2_handler_java>`.

Puede utilizar un manejador SOAP 1.1 o SOAP 1.2 en una interconexión de solicitante de servicio, pero en este caso, el manejador SOAP 1.2 no soporta los mensajes SOAP 1.1. No especifique el manejador SOAP 1.1 o SOAP 1.2 en la interconexión si las aplicaciones de solicitante de servicio están enviando sobres SOAP completos en el contenedor DFHREQUEST. Esto impide la duplicación de cabeceras de mensajes SOAP en mensajes de salida.

En un proveedor de servicios, puede especificar el manejador genérico y los manejadores SOAP en el elemento `<terminal_handler>` así como en el elemento `<service_handler_list>`. Para obtener más información sobre el procesamiento de cabeceras SOAP, consulte "Programas de proceso de cabeceras" en la página 145.

Ejemplo

```
<service_handler_list>
  <wsse_handler>
    ...
  </wsse_handler>
  <cics_soap_1.1_handler_java>
    ...
  </cics_soap_1.1_handler_java>
  <handler>
    ...
  </handler>
</service_handler_list>
```

Elemento `<service_parameter_list>`:

Especifica elementos XML que pasan a estar disponibles a todos los manejadores de mensajes en la interconexión en el contenedor DFH-SERVICEPLIST. Es un elemento opcional.

Se usa en lo siguiente:

- Solicitante de servicios
- Proveedor de servicios

Contiene:

- Si está utilizando WS-AT: un elemento `<registration_service_endpoint>`
- En un solicitante de servicio si está utilizando WS-AT: un elemento `<new_tx_context_required/>` opcional

- Etiquetas definidas por el usuario opcionales

Ejemplo

```
<requester_pipeline>
  <service_parameter_list>
    <registration_service_endpoint>
      http://provider.example.com:7160/cicswsat/RegistrationService
    </registration_service_endpoint>
    <new_tx_context_required/>
    <user_defined_tag1>
      ...
    </user_defined_tag1>
  </service_parameter_list>
</requester_pipeline>
```

Elemento <transport>:

Especifica los manejadores que se van a invocar sólo cuando está en uso un transporte particular.

Se usa en lo siguiente:

- Proveedor de servicios
- Solicitante de servicio

Contenido en:

```
<provider_pipeline>
<requester_pipeline>
```

Contiene:

En un proveedor de servicios

1. Un elemento <default_transport_handler_list> opcional
2. Un elemento <default_http_transport_handler_list> opcional
3. Un elemento <default_mq_transport_handler_list> opcional
4. Cero, uno o más elementos <named_transport_entry>

En un solicitante de servicio:

1. Un elemento <default_target> opcional. <default_target> contiene un URI que utiliza CICS para localizar el servicio web de destino cuando la aplicación de solicitante de servicio no proporciona un URI. Sin embargo, en muchos casos, la aplicación de solicitante de servicio proporcionará el URI del destino, e independientemente de lo que especifique en <default_target> se ignorará. Por ejemplo, las aplicaciones de proveedor de servicios que se despliegan utilizando el asistente de servicios web de CICS normalmente obtienen el URI de la descripción de servicio web.
2. Un elemento <default_http_transport_handler_list> opcional
3. Un elemento <default_mq_transport_handler_list> opcional
4. Un elemento <default_transport_handler_list> opcional

Ejemplo

```
<transport>
  <default_transport_handler_list>
    ...
  </default_transport_handler_list>
</transport>
```

Configuración de interconexión para MTOM/XOP

Las interconexiones CICS SOAP pueden soportar las especificaciones del Mecanismo de optimización de transmisión de mensajes (MTOM) y de Empaquetado optimizado binario XML (XOP). Estas especificaciones definen un mecanismo para enviar y recibir datos binarios utilizando SOAP, sin incurrir en la sobrecarga de la codificación base64. Para habilitar el soporte MTOM, debe configurar las interconexiones en consecuencia.

Elemento `<mtom>`:

Habilita el soporte MTOM/XOP para interconexiones basadas en Java. Si se define este elemento en el archivo de configuración de interconexión, el soporte MTOM está habilitado para todos los mensajes de entrada y salida. Sin embargo, si este elemento no se especifica en el archivo de configuración de interconexión, el soporte MTOM se habilita únicamente para mensajes de entrada.

Se usa en lo siguiente:

- Proveedor de servicios
- Solicitante de servicios

Contenido en:

```
<cics_soap_1.1_handler_java>
<cics_soap_1.2_handler_java>
```

Para los archivos de configuración de interconexión de solicitante y proveedor, el elemento `<mtom>` debe definirse después del elemento `<addressing>` opcional y antes del elemento `<headerprogram>` opcional.

Ejemplo

Para una interconexión modalidad solicitante o proveedor, puede especificar:

```
<cics_soap_1.2_handler_java>
  <jvmserver>JVMSERV1</jvmserver>
  <addressing></addressing>
  <mtom></mtom>
  <headerprogram>
    <program_name>HDRPROG4</program_name>
    <namespace>http://mynamespace</namespace>
    <localname>myheaderblock</localname>
    <mandatory>true</mandatory>
  </headerprogram>
</cics_soap_1.2_handler_java>
```

Elemento `<cics_mtom_handler>`:

Habilita el programa de manejador MTOM proporcionado para interconexiones SOAP. Este programa proporciona soporte para mensajes relacionados/ de varias partes MTOM MIME que contienen documentos XOP y archivos adjuntos binarios. El soporte MTOM está habilitado para todos los mensajes de entrada recibidos en la interconexión, pero el soporte MTOM para mensajes de salida se habilita de forma condicional sujeto a opciones adicionales.

Se usa en lo siguiente:

- Proveedor de servicios
- Solicitante de servicio

Contenido en:

```
<provider_pipeline>
  <requester_pipeline>
```

En un archivo de configuración de interconexión de proveedor, el elemento `<cics_mtom_handler>` se debe definir antes del elemento `<transport>`. En tiempo de ejecución, el programa del manejador MTOM necesita desempaquetar el mensaje MTOM de entrada antes de que otros manejadores, incluyendo el manejador de transporte, lo procesen. Es entonces cuando se invoca como el último manejador para el mensaje de respuesta, para empaquetar un mensaje MTOM para enviarlo al solicitante de servicio web.

En un archivo de configuración de interconexión de solicitante, el elemento `<cics_mtom_handler>` debe definirse después del elemento `<transport>`. En tiempo de ejecución, el mensaje de solicitud de salida no se convierte a formato MTOM hasta que todos los demás manejadores lo han procesado. Entonces se invoca como el primer manejador para el mensaje de respuesta de entrada para desempaquetar el mensaje MTOM antes de que otros manejadores lo procesen y vuelva al programa solicitante.

Nota: No debe utilizar este programa de manejador con interconexiones basadas en Java. Para interconexiones basadas en Java, especifique el elemento `<mtom>`.

Contiene:

Elemento `<dfhmtom_configuration>`

Las opciones predeterminadas se pueden cambiar utilizando opciones de configuración especificadas en el elemento `<dfhmtom_configuration>`. Si no desea cambiar las opciones predeterminadas, puede utilizar un elemento vacío.

Ejemplo

Para una interconexión de modalidad de proveedor, puede especificar:

```
<provider_pipeline>
  <cics_mtom_handler></cics_mtom_handler>
  <transport>
    ....
  </transport>
  <service>
    ....
  </service>
</provider_pipeline>
```

Elemento `<dfhmtom_configuration>`:

Especifica la información de configuración para el programa de manejador MTOM proporcionado para interconexiones que no soportan Java. Este programa proporciona soporte para mensajes MIME que contienen archivos adjuntos binarios y documentos XOP. Si no especifica ninguna configuración para MTOM, CICS presupone los valores predeterminados.

Se usa en lo siguiente:

- Proveedor de servicios
- Solicitante de servicio

Contenido en:

<cics_mtom_handler>

Atributos:

Nombre	Descripción
versión	Un entero que indica la versión de la información de configuración. El único valor válido es 1.

Contiene:

- Un elemento <mtom_options> opcional
- Un elemento <xop_options>
- Un elemento <mime_options> opcional

Ejemplo

```
<dfhmtom_configuration version="1">  
  <mtom_options send_mtom="same" send_when_no_xop="no"/>  
  <xop_options apphandler_supports_xop="yes"/>  
  <mime_options content_id_domain="example.org"/>  
</dfhmtom_configuration>
```

Elemento <mtom_options>:

Especifica cuánto utilizar MTOM para mensajes SOAP de salida para interconexiones que no soportan Java.

Se usa en lo siguiente:

- Proveedor de servicios
- Solicitante de servicios

Contenido en:

<dfhmtom_configuration>

Atributos:

Atributo	Descripción
send_mtom	<p>Especifica si se debe utilizar MTOM para convertir el mensaje SOAP de salida en un mensaje MIME:</p> <p>no MTOM no se utiliza para mensajes SOAP de salida.</p> <p>mismo En modalidad de proveedor de servicios, se utiliza MTOM para mensajes de respuestas SOAP siempre que el solicitante utiliza MTOM. Este es el valor predeterminado en una interconexión de proveedor de servicios.</p> <p>En modalidad de solicitante de servicio, la especificación de este valor equivale a especificar send_mtom="yes".</p> <p>yes Se utiliza MTOM para todos los mensajes SOAP de salida. Este es el valor predeterminado en una interconexión de solicitante de servicio.</p>

Atributo	Descripción
send_when_no_xop	<p>Especifica si se debe enviar un mensaje MTOM, incluso cuando no haya archivos adjuntos binarios presentes en el mensaje.</p> <p>no Se utiliza MTOM cuando se están enviando archivos adjuntos binarios con el mensaje.</p> <p>yes Se utiliza MTOM para todos los mensajes SOAP de salida, incluso cuando no haya archivos adjuntos binarios para enviar en el mensaje. Este es el valor predeterminado y se utiliza principalmente como un indicador al programa de recepción de que el remitente da soporte a MTOM/XOP.</p> <p>Este atributo se puede combinar con cualquiera de los valores de atributo send_mtom, pero no tiene ningún efecto si especifica send_mtom="no".</p>

Ejemplo

```

<provider_pipeline>
  <cics_mtom_handler>
    <dfhmtom_configuration version="1">
      <mtom_options send_mtom="same" send_when_no_xop="no"/>
    </dfhmtom_configuration>
  </cics_mtom_handler>
  ...
</provider_pipeline>

```

En este ejemplo de interconexión de proveedor, los mensajes SOAP se convierten en mensajes MTOM sólo cuando es necesario enviar archivos adjuntos binarios con el mensaje, y el solicitante de servicio ha enviado un mensaje MTOM.

Elemento <xop_options>:

Especifica si el proceso XOP puede tener lugar en modalidad de compatibilidad o directa para interconexiones que no soportan Java.

Se usa en lo siguiente:

- Proveedor de servicios
- Solicitante de servicios

Contenido en:

```
<dfhmtom_configuration>
```

Atributos:

Atributo	Descripción
apphandler_supports_xop	<p>En modalidad de proveedor, especifica si el manejador de aplicaciones es capaz de manejar documentos XOP en modalidad directa:</p> <p>no El manejador de aplicaciones no puede manejar documentos XOP directamente. Este es el valor predeterminado si el elemento <apphandler> no especifica DFHPITP.</p> <p>La modalidad de compatibilidad se utiliza en la interconexión para manejar cualquier mensaje de entrada o salida que se reciba o envíe en formato MTOM.</p> <p>yes El manejador de aplicaciones puede manejar documentos XOP. Es el valor predeterminado si el elemento <apphandler> especifica DFHPITP.</p> <p>La modalidad directa se utiliza en la interconexión para manejar cualquier mensaje de entrada o salida que se reciba o envíe en formato MTOM. Está sujeto a restricciones en tiempo de ejecución. Por ejemplo, si ha especificado elementos relacionados con WS-Security en el archivo de configuración de interconexión, el manejador MTOM determina que la interconexión debe utilizar modalidad de compatibilidad en lugar de modalidad directa para el proceso de los documentos XOP.</p> <p>En la modalidad de solicitante, especifica si las aplicaciones de solicitante de servicio utilizan el soporte de servicios web CICS para crear y gestionar documentos XOP en modalidad directa.</p> <p>no Las aplicaciones de solicitante de servicio no utilizan el soporte de servicios web CICS. Especifique este valor si la aplicación de solicitante enlaza con DFHPIRT para dirigir la interconexión y por lo tanto no es capaz de crear y manejar documentos XOP en modalidad directa.</p> <p>yes Las aplicaciones de solicitante de servicio utilizan el soporte de servicios web CICS. Especifique este valor si la aplicación de solicitante utiliza el mandato EXEC CICS INVOKE WEBSERVICE.</p>

Ejemplo

```
<provider_pipeline>
  <cics_mtom_handler>
    <dfhmtom_configuration version="1">
      <xop_options apphandler_supports_xop="no"/>
    </dfhmtom_configuration>
  </cics_mtom_handler>
  ...
</provider_pipeline>
```

En este ejemplo de interconexión de proveedor, los mensajes MTOM de entrada y los mensajes de respuesta de salida se procesan en la interconexión utilizando modalidad de compatibilidad.

Elemento <mime_options>:

Especifica el nombre de dominio que debe utilizarse cuando se generan los valores ID-contenido de MIME para interconexiones que no soportan Java. Los valores ID-contenido de MIME se utilizan para identificar archivos adjuntos binarios.

Se usa en lo siguiente:

- Proveedor de servicios
- Solicitante de servicios

Contenido en:

<dfhmtom_configuration>

Atributos:

Atributo	Descripción
content_id_domain	<p>La sintaxis a utilizar es <i>dominio.nombre</i>.</p> <p>Para adecuarse a los estándares de Internet, el nombre debe ser un nombre de host de Internet válido y debe ser exclusivo en el sistema CICS donde se instala la interconexión. Tenga en cuenta que CICS no lo comprueba.</p> <p>Si se omite este elemento, CICS utiliza el valor <i>cicsts</i>.</p>

Ejemplo

```
<provider_pipeline>
<dfhmtom_configuration version="1">
  <mime_options content_id_domain="example.org"/>
</dfhmtom_configuration>
...
</provider_pipeline>
```

En este ejemplo, las referencias a los archivos adjuntos binarios se crean utilizando *cid:valor_exclusivo@ejemplo.org*.

Configuración de interconexión para WS-Security

Para que las aplicaciones de proveedor y solicitante de servicio web participen en los protocolos WS-Security, debe configurar las interconexiones según corresponda, incluido el manejador de mensajes DFHWSSE, y proporcionando información de configuración para el manejador.

Ejemplo

Un archivo de configuración de interconexión de proveedor que utiliza WS-Security puede tener la forma siguiente:

```
<?xml version="1.0"?>
<provider_pipeline
  xmlns="http://www.ibm.com/software/http/cics/pipeline">
  <service>
    <service_handler_list>
      <wsse_handler>
        <dfhwsse_configuration version="1">
          <authentication trust="blind" mode="basic"/>
        </dfhwsse_configuration>
      </wsse_handler>
    </service_handler_list>
  </service>
  ...
</provider_pipeline>
```

```

        </handler>
      </service_handler_list>
      <terminal_handler>
        <cics_soap_1.2_handler/>
      </terminal_handler>
    </service>
    <apphandler>DFHPITP</apphandler>
  </provider_pipeline>

```

Elemento **<wsse_handler>**:

Especifica los parámetros utilizados por el manejador de mensajes proporcionado por CICS que proporciona soporte para WS-Security.

Se usa en lo siguiente:

- Proveedor de servicios
- Solicitante de servicio

Contenido en:

```
<service_handler_list>
```

Contiene:

- Un elemento `<dfhwsse_configuration>`.

En un archivo de configuración de interconexión de proveedor, el manejador de mensajes proporcionado por CICS para WS-Security puede que tenga que descifrar un mensaje cifrado. El elemento `<wsse_handler>` debe definirse antes de cualquier otro programa de manejador que necesite procesar el contenido del mensaje no cifrado.

En un archivo de configuración de interconexión de proveedor, el manejador de mensajes proporcionado por CICS para WS-Security puede que tenga que cifrar un mensaje cifrado. Debe definirse después de cualquier otro programa de manejador que necesite procesar el contenido del mensaje no cifrado, incluido el programa de manejador CICS SOAP.

Elemento **<dfhwsse_configuration>**:

Especifica la información de configuración para el manejador de seguridad DFHWSSE1, que proporciona soporte para proteger los servicios web.

Se usa en lo siguiente:

- Proveedor de servicios
- Solicitante de servicio

Contenido en:

```
<wsse_handler>
```

Atributos:

Nombre	Descripción
versión	Un entero que indica la versión de la información de configuración. El único valor válido es 1.

Contiene:**1. Uno de los elementos siguientes:**

- Un elemento `<authentication>` opcional.
 - En una interconexión de solicitante de servicio, el elemento `<authentication>` especifica el tipo de autenticación que debe utilizarse en el manejador de seguridad de los mensajes SOAP de salida.
 - En una interconexión de proveedor de servicios, el elemento especifica si CICS utiliza señales de seguridad en un mensaje SOAP de entrada para determinar el ID de usuario bajo el que se procesa el trabajo.
- Un elemento `<sts_authentication>` opcional.

El atributo `action` en este elemento especifica qué tipo de solicitud enviar al Security Token Service. Si la solicitud emitirá una señal de seguridad, CICS utiliza los valores en los elementos anidados para solicitar una señal de identidad del tipo especificado.

2. Si especifica un elemento `<sts_authentication>`, también debe especificar un elemento `<sts_endpoint>`.

Cuando este elemento está presente, CICS utiliza el URI en el elemento `<endpoint>` para enviar una solicitud al Servicio de señales de seguridad.

3. Un elemento `<expect_signed_body/>` vacío opcional.

El elemento `<expect_signed_body/>` indica que el cuerpo (`<body>`) del mensaje de entrada debe estar firmado. Si el cuerpo del mensaje de entrada no está correctamente firmado, CICS rechaza el mensaje con un error de seguridad.

4. Un elemento `<expect_encrypted_body/>` vacío opcional.

El elemento `<expect_encrypted_body/>` indica que el `<body>` del mensaje de entrada debe estar cifrado. Si el cuerpo del mensaje de entrada no está correctamente cifrado, CICS rechaza el mensaje con un error de seguridad.

5. Un elemento `<sign_body>` opcional.

Si este elemento está presente, CICS firmará el cuerpo (`<body>`) del mensaje de salida, utilizando el algoritmo especificado en el elemento `<algorithm>` contenido en el elemento `<sign_body>`.

6. Un elemento `<encrypt_body>` opcional.

Si este elemento está presente, CICS cifrará el cuerpo (`<body>`) del mensaje de salida, utilizando el algoritmo especificado en el elemento `<algorithm>` contenido en el elemento `<encrypt_body>`.

7. Sólo en las interconexiones del proveedor, un elemento `<reject_signature/>` opcional.

Si este elemento está presente, CICS rechaza los mensajes que incluyen un certificado en su cabecera que representa una parte o el total del cuerpo del mensaje. Se emite un error SOAP para el solicitante de servicio web.

8. Sólo en las interconexiones del proveedor, un elemento `<reject_encryption/>` opcional.

Si este elemento está presente, CICS rechaza cualquier mensaje que está cifrado parcial o totalmente. Se emite un error SOAP para el solicitante de servicio web.

Ejemplo

```
<dfhwsse_configuration version="1">
  <sts_authentication action="issue">
    <auth_token_type>
      <namespace>http://example.org.tokens</namespace>
      <element>UsernameToken</element>
    </auth_token_type>
  </sts_authentication>
</dfhwsse_configuration>
```

```

    <suppress/>
  </sts_authentication>
  <sts_endpoint>
    <endpoint>https://example.com/SecurityTokenService</endpoint>
  </sts_endpoint>
  <expect_signed_body/>
  <expect_encrypted_body/>
  <sign_body>
    <algorithm>http://www.w3.org/2000/09/xmldsig#rsa-sha1</algorithm>
    <certificate_label>SIGCERT01</certificate_label>
  </sign_body>
  <encrypt_body>
    <algorithm>http://www.w3.org/2001/04/xmlenc#tripledes-cbc</algorithm>
    <certificate_label>ENCCERT02</certificate_label>
  </encrypt_body>
</dfwsse_configuration>

```

El elemento <authentication>:

Especifica la utilización de las señales de seguridad en las cabeceras de los mensajes SOAP de entrada y de salida.

Se usa en lo siguiente:

- Proveedor de servicios
- Solicitante de servicio

Contenido en:

<dfwsse_configuration>

Atributos:

Atributo	Descripción
confianza y modalidad	<p>Si se toman conjuntamente, los atributos trust y mode especifican lo siguiente:</p> <ul style="list-style-type: none"> • si se utiliza identidad afirmada • la combinación de señales de seguridad utilizadas en mensajes SOAP. <p>La identidad certificada permite a un usuario de confianza certificar qué trabajo debe ejecutarse bajo una identidad diferente, la <i>identidad certificada</i>, sin que el usuario de confianza tenga las credenciales que están asociadas a esa identidad.</p> <p>Cuando se utiliza la identidad afirmada, los mensajes contienen una <i>señal de fiabilidad</i> y una <i>señal de identidad</i>. La señal de confianza se utiliza para comprobar que el remitente tiene los permisos correctos para certificar identidades. La señal de identidad contiene la identidad certificada, es decir, el ID de usuario bajo el que se ejecuta la solicitud.</p> <p>La utilización de la identidad afirmada requiere que un proveedor de servicios confíe en un solicitante para realizar su imposición. En CICS, la relación de fiabilidad se establece con las definiciones de sustitución del gestor de seguridad: la identidad solicitante debe tener la autoridad correcta para iniciar trabajo en nombre de la autoridad afirmada.</p> <p>Las combinaciones permitidas de estos atributos y sus significados se describen en Tabla 3 en la página 125 y Tabla 4 en la página 125.</p>

Tabla 3. Atributos **mode** y **trust** en una interconexión de solicitante de servicio

trust	modalidad	Significado
ninguno	ninguno	No se añaden credenciales al mensaje
ninguno	básico	<i>Combinación no válida de valores de atributo</i>
ninguno	firma	No se utiliza la identidad afirmada. CICS utiliza una señal de seguridad X.509 única, que se añade al mensaje, y se utiliza para firmar el cuerpo del mensaje. El certificado se identifica con el elemento <certificate_label> y el algoritmo se especifica en el elemento <algorithm>.
blind	ninguno	<i>Combinación no válida de valores de atributo</i>
blind	básico	No se utiliza la identidad certificada. CICS añade una señal de identidad al mensaje, pero no proporciona una señal de confianza. La señal de identidad es un nombre de usuario sin contraseña. El ID de usuario colocado en la señal de identidad es el contenido del contenedor DFHWS-USERID (que, de manera predeterminada, contiene el ID de usuario de la tarea en ejecución).
blind	firma	<i>Combinación no válida de valores de atributo</i>
básico	Ninguno	<i>Combinación no válida de valores de atributo</i>
básico	básico	<i>Combinación no válida de valores de atributo</i>
básico	firma	<i>Combinación no válida de valores de atributo</i>
firma	Ninguno	<i>Combinación no válida de valores de atributo</i>
firma	básico	Se utiliza la identidad afirmada. CICS añade las señales siguientes al mensaje: <ul style="list-style-type: none"> • La señal de fiabilidad es una señal de seguridad X.509. • La señal de identidad es un nombre de usuario sin contraseña. El certificado que se utiliza para firmar la señal de identidad y el cuerpo del mensaje se especifica mediante <certificate_label>. El ID de usuario colocado en la señal de identidad es el contenido del contenedor DFHWS-USERID (que, de manera predeterminada, contiene el ID de usuario de la tarea en ejecución).
firma	firma	<i>Combinación no válida de valores de atributo</i>

Tabla 4. Atributos **mode** y **trust** en una interconexión de proveedor de servicios

de confianza	modalidad	Significado
ninguno	ninguno	Es necesario que los mensajes de entrada no contengan ninguna credencial y que CICS no intente extraer ni verificar las credenciales que se encuentran en un mensaje. Sin embargo, CICS comprueba que los elementos firmados están correctamente firmados.
ninguno	básico	Los mensajes de entrada deben contener una señal de seguridad de nombre de usuario con una contraseña. CICS pone el nombre de usuario en el contenedor DFHWS-USERID.

Tabla 4. Atributos **mode** y **trust** en una interconexión de proveedor de servicios (continuación)

de confianza	modalidad	Significado
ninguno	ICRX básico	<i>Combinación no válida de valores de atributo</i>
Ninguno	kerberos básico	<i>Combinación no válida de valores de atributo</i>
ninguno	firma	Los mensajes de entrada deben contener una señal de seguridad X.509 que se ha utilizado para firmar el cuerpo del mensaje.
blind	ninguno	<i>Combinación no válida de valores de atributo</i>
blind	básico	Los mensajes de entrada deben contener una señal de identidad, donde la señal de identidad contiene un ID de usuario y, opcionalmente, una contraseña. CICS pone el ID de usuario en el contenedor DFHWS-USERID. Si no se incluye ninguna contraseña, CICS utiliza el ID de usuario sin verificarlo. Si se incluye una contraseña, el manejador de seguridad DFHWSSE1 la verifica.
blind	ICRX básico	Los mensajes de entrada deben contener una señal de identidad ICRX. CICS resuelve la identidad, pone el ID de usuario en el contenedor DFHWS-USERID y al ICRX en el contenedor DFHWS-ICRX. La autenticación, si es necesaria, utiliza el SSL certificado por el cliente u otro protocolo de seguridad.
blind	kerberos básico	<i>Combinación no válida de valores de atributo</i>
blind	firma	Los mensajes de entrada deben contener una señal de identidad, donde la señal de identidad es el primer certificado X.509 en la cabecera del mensaje SOAP. No es necesario que el certificado haya firmado el mensaje. El manejador de seguridad extrae el ID de usuario coincidente y lo coloca en el contenedor DFHWS-USERID.
básico	ninguno	<i>Combinación no válida de valores de atributo</i>
básico	básico	Los mensajes de entrada deben utilizar identidad afirmada: <ul style="list-style-type: none"> La señal de confianza es una señal de nombre de usuario con una contraseña La señal de identidad es una segunda señal de nombre de usuario sin una contraseña. CICS pone este nombre de usuario en el contenedor DFHWS-USERID.
básico	ICRX básico	Los mensajes de entrada deben utilizar identidad afirmada: <ul style="list-style-type: none"> La señal de confianza es una señal de nombre de usuario con una contraseña <p>CICS establece si la combinación de ID de usuario y contraseña es válida y, si lo es, CICS resuelve la identidad certificada basada en el ICRX para un ID de usuario. CICS realiza una comprobación de seguridad sustituta desde la identidad autenticada a la identidad certificada.</p> <ul style="list-style-type: none"> La señal de identidad es un ICRX, que identifica el usuario cliente específico. CICS pone el nombre de usuario en el contenedor DFHWS-USERID y el ICRX en el contenedor DFHWS-ICRX.

Tabla 4. Atributos **mode** y **trust** en una interconexión de proveedor de servicios (continuación)

de confianza	modalidad	Significado
básico	kerberos básico	<p>Los mensajes de entrada deben utilizar identidad certificada.</p> <p>Se necesita una señal, un señal Kerberos Versión 5 con uno de los siguientes tipos de formato:</p> <ul style="list-style-type: none"> • http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#Kerberosv5_AP_REQ1510 • http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#GSS_Kerberosv5_AP_REQ1510 • http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#Kerberosv5_AP_REQ4120 • http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#GSS_Kerberosv5_AP_REQ4120 <p>La señal debe tener la codificación Base-64. CICS valida la señal utilizando el Servicio de autenticación de red para z/OS y pone el ID de usuario asociado a la señal en el contenedor DFHWS-USERID.</p>
básico	firma	<p>Los mensajes de entrada deben utilizar identidad afirmada:</p> <ul style="list-style-type: none"> • La señal de confianza es una señal de nombre de usuario con una contraseña • La señal de identidad es un certificado X.509. CICS pone el ID de usuario asociado con el certificado en el contenedor DFHWS-USERID.
firma	ninguno	<i>Combinación no válida de valores de atributo</i>
firma	básico	<p>Los mensajes de entrada deben utilizar identidad afirmada:</p> <ul style="list-style-type: none"> • La señal de fiabilidad es un certificado X.509 • La señal de identidad es una segunda señal de nombre de usuario sin una contraseña. CICS pone este nombre de usuario en el contenedor DFHWS-USERID. <p>La señal de identidad y el cuerpo deben estar firmados con el certificado X.509.</p>

Tabla 4. Atributos **mode** y **trust** en una interconexión de proveedor de servicios (continuación)

de confianza	modalidad	Significado
firma	ICRX básico	<p>Los mensajes de entrada deben utilizar identidad certificada.</p> <ul style="list-style-type: none"> La señal de confianza es un ICRX firmado con un certificado X.509. <p>CICS resuelve el certificado X.509 para un ID de usuario y asegura que la firma XML es válida. CICS resuelve la identidad certificada basada en ICRX para un ID de usuario. CICS realiza una comprobación de seguridad sustituta desde la identidad X.509 autenticada a la identidad ICRX certificada.</p> <ul style="list-style-type: none"> La señal de identidad es una segunda señal de nombre de usuario sin una contraseña. CICS pone el nombre de usuario en el contenedor DFHWS-USERID y el ICRX en el contenedor DFHWS-ICRX.
firma	kerberos básico	<i>Combinación no válida de valores de atributo</i>
firma	firma	<p>Los mensajes de entrada deben utilizar identidad afirmada:</p> <ul style="list-style-type: none"> La señal de fiabilidad es un certificado X.509 La señal de identidad es un segundo certificado X.509. CICS pone el ID de usuario asociado con este certificado en el contenedor DFHWS-USERID. <p>La señal de identidad y el cuerpo deben estar firmados con el primer certificado X.509 (la señal de fiabilidad).</p>

Notas:

1. Las combinaciones de los atributos trust y mode se comprueban cuando se instala PIPELINE. La instalación falla si los atributos están codificados de forma incorrecta.
2. CICS utiliza la verificación de contraseña para verificar un ID de usuario durante los procesos descritos en VERIFY PHRASE.

Contiene:

1. Un elemento <suppress/> vacío opcional.

Si este elemento se especifica en una interconexión de proveedor de servicios, el manejador no intenta utilizar ninguna señal de seguridad en el mensaje para determinar bajo qué ID de usuario se ejecuta el trabajo.

Si este elemento se especifica en una interconexión de solicitante de servicio, el manejador no intenta añadir al mensaje SOAP de salida ninguna señal de seguridad necesaria para la autenticación.
2. En una interconexión de solicitante, un elemento <algorithm> opcional que especifica el URI del algoritmo que se utiliza para firmar el cuerpo del mensaje SOAP. Debe especificar este elemento si la combinación de los valores de los atributos trust y mode indican que los mensajes están firmados. Sólo puede especificar el RSA con el algoritmo SHA1 en este elemento. El URI es <http://www.w3.org/2000/09/xmldsig#rsa-sha1>.
3. Un elemento <certificate_label> opcional que especifica la etiqueta asociada a un certificado digital X.509 instalado en RACF. Si especifica este elemento en

una interconexión de solicitante de servicio y el elemento <suppress> no se especifica, el certificado se añade a la cabecera de seguridad en el mensaje SOAP. Si no especifica un elemento <certificate_label>, CICS utiliza el certificado predeterminado en el anillo de claves RACF.

Este elemento se ignora en una interconexión de proveedor de servicios.

Ejemplo

```
<authentication
trust="signature" mode="basic">
  <suppress/>
  <algorithm>http://www.w3.org/2000/09/xmldsig#rsa-sha1</algorithm>
  <certificate_label>AUTHCERT03</certificate_label>
</authentication>
```

Elemento <sts_authentication>:

Especifica que se debe utilizar un Security Token Service (STS) para la autenticación y determina qué tipo de solicitud se envía.

Se usa en lo siguiente:

- Proveedor de servicios
- Solicitante de servicios

Contenido en:

```
<dfhwsse_configuration>
```

Atributos:

Nombre	Descripción
acción	<p>Especifica qué tipo de solicitud envía CICS al STS cuando se recibe un mensaje en la interconexión de proveedor de servicios. Los valores válidos son los siguientes:</p> <p>issue El STS emite una señal de identidad para el mensaje SOAP. Este valor no es válido para SAML en una interconexión de proveedor.</p> <p>validate El STS valida la señal de identidad proporcionada y devuelve si la señal es válida al manejador de seguridad.</p> <p>**Si no especifica este atributo, CICS asume que la acción es para solicitar una señal de identidad.</p> <p>En una interconexión de solicitante de servicio, no puede especificar este atributo porque CICS siempre solicita que el STS emita una señal.</p>
extract	<p>Este atributo sólo es válido cuando está utilizando SAML. ¿Se van a extraer los elementos de la señal SAML? Los valores válidos son los siguientes:</p> <p>no Los elementos de la señal SAML no se van a extraer a contenedores.</p> <p>yes Los elementos principales de la señal SAML se extraen y colocan en contenedores creados por CICS.</p>

Nombre	Descripción
token_signature	<p>Este atributo sólo es válido cuando está utilizando SAML. ¿Se debe proporcionar una firma de señal? Los valores válidos son los siguientes:</p> <p>ignored Todas las firmas proporcionadas se ignoran.</p> <p>required Se debe proporcionar una firma válida. Este es el valor predeterminado.</p>
tran_channel	<p>Este atributo sólo es válido cuando está utilizando SAML. En una interconexión de proveedor de servicios, este atributo especifica si las aserciones SAML contenidas en un mensaje recibido en la interconexión están disponibles para el programa de aplicación de destino en contenedores del canal de transacción DFHTRANSACTION. Los valores válidos son los siguientes:</p> <p>yes Las aserciones SAML se copian en contenedores en el canal DFHTRANSACTION para que estén disponibles para el programa. Para obtener más información sobre los nombres y tipos de contenedor, consulte .</p> <p>no Las aserciones SAML no están disponibles para el programa a través del canal DFHTRANSACTION, sino en contenedores en el canal que pasan al programa a través de la interconexión. Este es el valor predeterminado.</p> <p>Si no especifica este atributo para un proveedor de servicios, las aserciones sólo están disponibles en contenedores en el canal que se pasa al programa desde la interconexión SOAP.</p> <p>En una interconexión de solicitante de servicio, este atributo especifica si la señal SAML contenida en el contenedor DFHSAML-OUTTOKEN del canal de transacción DFHTRANSACTION se utiliza en la solicitud. Los valores válidos son los siguientes:</p> <p>yes El contenido del contenedor DFHSAML-OUTTOKEN del canal DFHTRANSACTION se utiliza como señal SAML para la solicitud.</p> <p>no El contenido del contenedor DFHSAML-OUTTOKEN en el canal que pasa a la interconexión se utiliza como señal SAML de la solicitud. Este es el valor predeterminado.</p> <p>Si no especifica este atributo para un solicitante de servicio, la señal SAML se toma del contenedor DFHSAML-OUTTOKEN en el canal que pasa a la interconexión SOAP.</p>

Contiene:

1. Un elemento <auth_token_type>. Este elemento es obligatorio cuando especifica un elemento <sts_authentication> en una interconexión de solicitante de servicio y es opcional en un proveedor de servicios. Para obtener más información, consulte <auth_token_type>.

- En una interconexión de solicitante de servicio, el elemento `<auth_token_type>` indica el tipo de señal que emite STS cuando CICS la envía al ID de usuario contenido en el contenedor DFHWS-USERID. La señal que recibe CICS desde el STS se coloca en la cabecera del mensaje de salida.
 - En una interconexión de proveedor de servicios, el elemento `<auth_token_type>` se utiliza para determinar la señal de identidad que toma CICS de la cabecera de mensaje y la envía al STS para intercambiar o validar. CICS utiliza la primera señal de identidad del tipo especificado en la cabecera de mensaje. Si no especifica este elemento, CICS utiliza la primera señal de identidad que se encuentra en la cabecera de mensaje. CICS no considera los siguientes elementos como señales de identidad:
 - `wsu:Timestamp`
 - `xenc:ReferenceList`
 - `xenc:EncryptedKey`
 - `ds:Signature`
2. En una interconexión de proveedor de servicios solamente, un elemento `<suppress/>` vacío opcional. Si se especifica este elemento, el manejador no intenta utilizar señales de seguridad en el mensaje para determinar el ID de usuario bajo el que se ejecuta el trabajo. El elemento `<suppress/>` incluye la señal de identidad que devuelve STS.

Ejemplo

El ejemplo siguiente muestra una interconexión de proveedor de servicios, donde el manejador de seguridad solicita una señal del STS.

```
<sts_authentication action="issue">
  <auth_token_type>
    <namespace>http://example.org.tokens</namespace>
    <element>UsernameToken</element>
  </auth_token_type>
  <suppress/>
</sts_authentication>
```

Elemento `<auth_token_type>`:

Especifica qué tipo de señal de identidad se requiere.

Este elemento es obligatorio cuando especifica el elemento `<sts_authentication>` en una interconexión de solicitante de servicio y opcional en un proveedor de servicios.

- En una interconexión de solicitante de servicio, el elemento `<auth_token_type>` indica el tipo de señal que emite STS cuando CICS la envía al ID de usuario contenido en el contenedor DFHWS-USERID. La señal que recibe CICS desde el STS se coloca en la cabecera del mensaje de salida.
- En una interconexión de proveedor de servicios, el elemento `<auth_token_type>` se utiliza para determinar la señal de identidad que toma CICS de la cabecera de mensaje y la envía al STS para intercambiar o validar. CICS utiliza la primera señal de identidad del tipo especificado en la cabecera de mensaje. Si no especifica este elemento, CICS utiliza la primera señal de identidad que se encuentra en la cabecera de mensaje. CICS no considera los siguientes elementos como señales de identidad:
 - `wsu:Timestamp`
 - `xenc:ReferenceList`
 - `xenc:EncryptedKey`

- ds:Signature

Se usa en lo siguiente:

- Proveedor de servicios
- Solicitante de servicio

Contenido en:

<sts_authentication>

Contiene:

1. Un elemento <namespace>. Este elemento contiene el espacio de nombres del tipo de señal que se va a validar o intercambiar.
Si está utilizando SAML, establezca el contenido de este elemento en urn:oasis:names:tc:SAML:1.0:assertion o urn:oasis:names:tc:SAML:2.0:assertion, dependiendo de la versión de SAML.
2. Un elemento <element>. Este elemento contiene el nombre local del tipo de señal que se va a validar o intercambiar.
Para SAML, utilice el nombre local Aserción.

Los valores de estos elementos forman el Qname de la señal.

Ejemplo

```
<auth_token_type>
  <namespace>http://example.org.tokens</namespace>
  <element>UsernameToken</element>
</auth_token_type>
```

Elemento <sts_endpoint>:

Especifica la ubicación del Security Token Service (STS).

Se usa en lo siguiente:

- Proveedor de servicios
- Solicitante de servicios

Contenido en:

<dfwsse_configuration>

Contiene:

- Un elemento <endpoint>. Este elemento contiene un URI que apunta a la ubicación del servicio de señales de seguridad (STS) en la red. Se recomienda que utilice SSL o TLS para mantener la conexión al STS segura, en lugar de utilizar HTTP.
Para utilizar el soporte SAML, establezca el punto final en cics://PROGRAM/DFHSAML.
También puede especificar un punto final WebSphere MQ, utilizando el formato JMS de URI.
- Un elemento <jvmserver> opcional. Este elemento identifica el servidor JVM configurado para ejecutar el servicio de señal SAML. Si este elemento no está incluido, el servidor JVM del recurso de muestra predeterminado DFHXSTS se asume. Este elemento es válido sólo si está utilizando SAML: si lo utiliza en otras situaciones, se produce un error

Ejemplos

En este ejemplo, el punto final está configurado para utilizar una conexión segura al STS en el URI especificado.

```
<sts_endpoint>
  <endpoint>https://example.com/SecurityTokenService</endpoint>
</sts_endpoint>
```

En este ejemplo, el punto final está configurado para utilizar el soporte SAML de CICS.

```
<sts_endpoint>
  <endpoint>cics://PROGRAM/DFHSAML</endpoint>
</sts_endpoint>
```

Elemento **<sign_body>**:

Dirige a DFHWSSE para firmar los mensajes SOAP de salida y proporciona información sobre cómo se van a firmar los mensajes.

Se usa en lo siguiente:

- Proveedor de servicios
- Solicitante de servicios

Contenido en:

```
<dfhwsse_configuration>
```

Contiene:

1. Un elemento **<algorithm>** que contiene el URI que identifica el algoritmo utilizado para firmar el cuerpo del mensaje SOAP. Puede especificar los algoritmos mostrados en “Algoritmos de firma” en la página 612.
2. Un elemento **<certificate_label>** que especifica la etiqueta asociada a un certificado digital instalado en RACF. El certificado digital proporciona la clave que se utiliza para firmar el mensaje.

Ejemplo

```
<sign_body>
  <algorithm>http://www.w3.org/2000/09/xmldsig#rsa-sha1</algorithm>
  <certificate_label>SIGCERT01</certificate_label>
</sign_body>
```

Elemento **<encrypt_body>**:

Dirige a DFHWSSE para cifrar el cuerpo de los mensajes SOAP de salida, y proporciona información sobre cómo se van a cifrar los mensajes.

Se usa en lo siguiente:

- Proveedor de servicios
- Solicitante de servicio

Contenido en:

```
<dfhwsse_configuration>
```

Contiene:

1. Un elemento `<algorithm>` que contiene el URI que identifica el algoritmo utilizado para cifrar el cuerpo del mensaje SOAP. Puede especificar los algoritmos mostrados en “Algoritmos de cifrado” en la página 614.
2. Un elemento `<certificate_label>` que especifica la etiqueta asociada a un certificado digital en RACF. El certificado digital proporciona la clave que se utiliza para cifrar el mensaje.

Ejemplo

```
<encrypt_body>  
  <algorithm>http://www.w3.org/2001/04/xmlenc#aes256-cbc</algorithm>  
  <certificate_label>ENCCERT02</certificate_label>  
</encrypt_body>
```

Manejadores de aplicación

Un manejador de aplicación es un programa CICS al que se enlaza el controlador de terminal de una interconexión de proveedor de servicio SOAP en el tiempo de ejecución.

Los manejadores de aplicación se utilizan en las interconexiones de modalidad de proveedor en las que el controlador de terminal es uno de los manejadores de mensajes SOAP proporcionados. Esta situación se produce cuando el elemento `<terminal_handler>` contiene un elemento `<cics_soap_1.1_handler>`, `<cics_soap_1.2_handler>`, `<cics_soap_1.1_handler_java>` o `<cics_soap_1.2_handler_java>`.

El manejador de aplicación es responsable de procesar el cuerpo de una solicitud SOAP y de generar una respuesta utilizando los datos devueltos. El manejador de aplicación puede invocar a otros programas para completar este proceso. Normalmente, los manejadores de aplicación actúan como una capa de presentación de propósito general alrededor de una o varias aplicaciones empresariales. Es responsable de la correlación de XML en un formato que puede utilizar una aplicación, de conectar esa aplicación y de generar después una respuesta utilizando los datos devueltos.

CICS puede conectar un manejador de aplicación de dos formas. El mecanismo típico implica un canal y contenedores de control; el otro método implica enlaces Java para Axis2.

Los manejadores de aplicación conectados por canal se especifican en el elemento `<apphandler>` del elemento `<provider_pipeline>`. En el tiempo de ejecución, el contenedor DFHWS-APPHANDLER se llena con el contenido de `<apphandler>`. Sin embargo, el contenedor DFHWS-APPHANDLER se puede actualizar dinámicamente por medio de cualquiera de los manejadores de mensajes. Por lo tanto, el programa con el que se enlaza en el tiempo de ejecución puede ser diferente al programa especificado en el elemento `<apphandler>`. Los siguientes manejadores de aplicación se pueden especificar en el elemento `<apphandler>` o el contenedor DFHWS-APPHANDLER:

- El manejador de aplicación SOAP conectado por canal proporcionado, DFHPITP. Para obtener más información sobre los manejadores de aplicación conectados por canal, consulte “Manejadores de aplicación conectados por canal” en la página 135
- Su propio manejador de aplicación conectado por canal. Este manejador de aplicación se puede escribir en lenguajes distintos a Java. Para obtener más

información sobre los contenedores de control que se pueden utilizar en el manejador de aplicación conectado por canal, consulte “Contenedores de control” en la página 152.

- Su propio manejador de aplicación Java para interconexiones basadas en Java, que implementa la interfaz de Java ApplicationHandler y que está conectado a la interconexión utilizando Axis2 MessageContext. Para obtener más información sobre la interfaz de Java ApplicationHandler, consulte JCICS class reference.

Para utilizar un manejador de aplicación que utiliza enlaces Java para Axis2, debe especificar el elemento `<apphandler_class>` del elemento `<provider_pipeline>`. Los manejadores de aplicación Axis2 también requieren que debe existir un servidor JVM para la interconexión de servicios web y el manejador de aplicación en el que ejecutarse y que el controlador de terminal de la interconexión de servicios web debe estar en el manejador de mensajes `<cics_soap_1.1_handler_java>` o `<cics_soap_1.2_handler_java>`. Para utilizar el manejador de aplicación Axis2 proporcionado, debe especificar `com.ibm.cicsts.axis2.CICSAxis2ApplicationHandler` en el elemento `<apphandler_class>`, sin embargo, puede especificar su propia clase de manejador de aplicación Axis2. En el tiempo de ejecución, el contenedor DFHWS-APPHANCLAS se llena con el contenido de `<apphandler_class>`.

Para aplicaciones de servicio web desplegadas utilizando el asistente de servicios web CICS, debe especificar DFHPITP o el manejador de aplicación que utiliza DFHPITP en el elemento `<apphandler>`, o especificar `com.ibm.cicsts.axis2.CICSAxis2ApplicationHandler` en el elemento `<apphandler_class>`. Para obtener más información sobre el asistente de servicios web CICS, consulte The CICS web services assistant.

También es posible desplegar aplicaciones Axis2 como servicios web de modalidad de proveedor en CICS utilizando el estilo Axis2 del despliegue de servicio web. Para obtener más información, consulte el apartado Deploying a Java provider-mode web service in an Axis2 JVM server.

Manejadores de aplicación conectados por canal

Los manejadores de aplicación conectados por canal son manejadores de aplicación que están conectados a CICS mediante un canal y contenedores de control.

El canal que utiliza el manejador de aplicación es el canal DFHAHC-V1. Este canal pasa los siguientes contenedores entre el controlador de terminal y la aplicación de servicio web modalidad de proveedor:

DFHWS-XMLNS

Contiene una lista de pares nombre-valor que correlaciona prefijos de espacio de nombres a espacios de nombres.

- En la entrada, la lista contiene los espacios de nombres que están en el ámbito desde el sobre SOAP.
- En la salida, la lista contiene los datos del espacio de nombres que se supone que están en la etiqueta del sobre.

DFHWS-BODY

Contiene la sección de cuerpo del sobre SOAP. Generalmente, la aplicación modificará el contenido. Si la aplicación no modifica el contenido, el programa de manejador de aplicación debe actualizar el contenido de este contenedor, aunque esté poniendo el mismo contenido de vuelta al contenedor antes de volver al controlador de terminal.

DFHNORESPONSE

En la fase de solicitud de una interconexión de solicitante de servicio, se indica que no se espera que el proveedor de servicios devuelva una respuesta. El contenido del contenedor DFHNORESPONSE está sin definir; los manejadores de mensajes que necesitan saber si se espera que el proveedor de servicios devuelva una respuesta, sólo necesitan determinar si el contenedor está presente o no:

- Si el contenedor DFHNORESPONSE está presente, no se espera una respuesta.
- Si el contenedor no está presente, se espera una respuesta.

El canal también pasa todos los contenedores de contexto que se pasaron al controlador de terminal. Por ejemplo, un programa de proceso de cabeceras puede añadir contenedores al canal. Estos contenedores se pasan como contenedores de usuario. Para obtener más información sobre los manejadores de aplicación, consulte “Manejadores de aplicación” en la página 92.

Manejadores de mensajes

Un manejador de mensajes es un programa de CICS que se utiliza para procesar una solicitud de servicio web durante la entrada y para procesar la respuesta durante la salida. Los manejadores de mensajes utilizan canales y contenedores para interactuar con entre ellos y con el sistema.

La interfaz del manejador de mensajes le permite realizar las siguientes tareas en un programa de manejador de mensajes:

- Examinar el contenido de una solicitud o respuesta XML o JSON, sin cambiarla
- Cambiar el contenido de una solicitud o respuesta XML o JSON
- En un manejador de mensajes de no terminal, pasar una solicitud o respuesta XML o JSON al siguiente manejador de mensajes en la interconexión
- En un manejador de mensajes de terminal, llamar a un programa de aplicación y generar una respuesta
- En la fase de solicitud de la interconexión, forzar una transición a la fase de respuesta, absorbiendo la solicitud y generando una respuesta
- Gestionar errores

Consejo: Es aconsejable utilizar los manejadores SOAP, `<cics_soap_1.1_handler>`, `<cics_soap_1.2_handler>`, `<cics_soap_1.1_handler_java>` o `<cics_soap_1.2_handler_java>`, para trabajar con mensajes SOAP. Estos manejadores le permiten trabajar directamente con los elementos principales en un mensaje SOAP (las cabeceras SOAP y el cuerpo SOAP).

Todos los programas utilizados como manejadores de mensajes se invocan con la misma interfaz: se invocan con un *canal* que contiene varios contenedores. Los contenedores se pueden categorizar como los tipos siguientes:

Contenedores de control

Son fundamentales para el funcionamiento de la interconexión. Los manejadores de mensajes pueden utilizar los contenedores de control para modificar la secuencia en la que se procesan los manejadores posteriores.

Contenedores de contexto

En algunas situaciones, los programas de manejadores de mensajes

necesitan información sobre el contexto en el que se invocan. CICS proporciona esta información en un conjunto de *contenedores de contexto* que se pasan a los programas.

Algunos de los contenedores de contexto contienen información que puede cambiar en el manejador de mensajes. Por ejemplo, en una interconexión de proveedor de servicios, puede cambiar el ID de usuario y el ID de transacción del programa de aplicación de destino modificando el contenido de los contenedores de contexto adecuados.

Contenedores de usuarios

Contienen la información que un manejador de mensajes necesita pasar a otro. El uso de los contenedores de usuarios es un asunto de los manejadores de mensajes.

Restricción: No utilice nombres que empiecen por DFH para los contenedores de usuarios.

Cómo controlan los contenedores los protocolos de interconexión

El contenido conjunto de los contenedores DFHFUNTION, DFHREQUEST y DFHRESPONSE controlan los protocolos de interconexión.

Durante las dos fases de ejecución de una interconexión (la fase de solicitud y la fase de respuesta), el valor de DFHFUNTION determina qué contenedores de control se pasan a cada manejador de mensajes:

DFHFUNTION	Contexto	DFHREQUEST	DFHRESPONSE
RECEIVE-REQUEST	Proveedor de servicios; fase de solicitud	Presente (longitud > 0)	Presente (longitud = 0)
SEND-RESPONSE	Proveedor de servicios; fase de respuesta	Ausente	Presente (longitud > 0)
SEND-REQUEST	Solicitante de servicio; fase de solicitud	Presente (longitud > 0)	Presente (longitud = 0)
RECEIVE-RESPONSE	Solicitante de servicio; fase de respuesta	Ausente	Presente (longitud > 0)
PROCESS-REQUEST	Proveedor de servicios; controlador de terminal	Presente (longitud > 0)	Presente (longitud = 0)
HANDLER-ERROR	Proveedor o solicitante de servicio; cualquier fase	Ausente	Presente (longitud = 0)
NO-RESPONSE	Proveedor o solicitante de servicio; fase de respuesta	Ausente	Ausente

El proceso subsiguiente se determina por los contenedores que el manejador de mensajes pasa de vuelta a la interconexión:

Durante la fase de la solicitud

- El manejador de mensajes puede devolver el contenedor DFHREQUEST. El proceso continúa en la fase de solicitud con el siguiente manejador. La longitud de los datos en el contenedor no debe ser cero.
- El manejador de mensajes puede devolver el contenedor DFHRESPONSE. El proceso conmuta a la fase de respuesta y se invoca al mismo manejador con DFHFUNCTIO establecido en SEND-RESPONSE en un proveedor de servicios y en RECEIVE-RESPONSE en un solicitante de servicio. La longitud de los datos en el contenedor no debe ser cero.
- El manejador de mensajes puede no devolver ningún contenedor. El proceso conmuta a la fase de respuesta y se invoca al mismo manejador con DFHFUNCTIO establecido en NO-RESPONSE.

En el controlador de terminal (sólo proveedor de servicios)

- El manejador de mensajes puede devolver el contenedor DFHRESPONSE. El proceso conmuta a la fase de respuesta y se invoca al manejador anterior con un nuevo valor de DFHFUNCTIO (SEND-RESPONSE). La longitud de los datos en el contenedor no debe ser cero.
- El manejador de mensajes puede no devolver ningún contenedor. El proceso conmuta a la fase de respuesta y se invoca al manejador anterior con un nuevo valor de DFHFUNCTIO (NO-RESPONSE).

Durante la fase de respuesta

- El manejador de mensajes puede devolver el contenedor DFHRESPONSE. El proceso continúa en la fase de respuesta y se invoca al siguiente manejador. La longitud de los datos en el contenedor no debe ser cero.
- El manejador de mensajes puede no devolver ningún contenedor. El proceso continúa en la fase de respuesta y se invoca al siguiente manejador en la secuencia con un nuevo valor de DFHFUNCTIO (NO-RESPONSE).

Importante: Durante la fase de solicitud, el manejador de mensajes puede devolver DFHREQUEST o DFHRESPONSE, pero no ambos. Dado que ambos contenedores están presentes cuando se invoca al manejador de mensajes, debe suprimir uno de ellos.

Esta tabla muestra la acción tomada por la interconexión para todos los valores de DFHFUNCTIO y todas las combinaciones de DFHREQUEST y DFHRESPONSE devueltas por cada manejador de mensajes.

DFHFUNCTIO	Contexto	DFHREQUEST	DFHRESPONSE	Acción
RECEIVE-REQUEST	Proveedor de servicios; fase de solicitud	Presente (longitud > 0)	Presente	(error)
RECEIVE-REQUEST	Proveedor de servicios; fase de solicitud	Presente (longitud > 0)	Ausente	Invoque al siguiente manejador con la función RECEIVE-REQUEST
RECEIVE-REQUEST	Proveedor de servicios; fase de solicitud	Presente (longitud = 0)	No aplicable	(error)
RECEIVE-REQUEST	Proveedor de servicios; fase de solicitud	Ausente	Presente (longitud > 0)	Conmute a la fase de respuesta e invoque el mismo manejador con la función SEND-RESPONSE

DFHFUNCTION	Contexto	DFHREQUEST	DFHRESPONSE	Acción
RECEIVE-REQUEST	Proveedor de servicios; fase de solicitud	Ausente	Presente (longitud = 0)	(error)
RECEIVE-REQUEST	Proveedor de servicios; fase de solicitud	Ausente	Ausente	Invoque al mismo manejador con la función NO-RESPONSE
SEND-RESPONSE	Proveedor de servicios; fase de respuesta	No aplicable	Presente (longitud > 0)	Invoque al manejador anterior con la función SEND-RESPONSE
SEND-RESPONSE	Proveedor de servicios; fase de respuesta	No aplicable	Presente (longitud = 0)	(error)
SEND-RESPONSE	Proveedor de servicios; fase de respuesta	No aplicable	Ausente	Invoque al mismo manejador con la función NO-RESPONSE
SEND-REQUEST	Solicitante de servicio; fase de solicitud	Presente (longitud > 0)	Presente (longitud = 0)	(error)
SEND-REQUEST	Solicitante de servicio; fase de solicitud	Presente (longitud > 0)	Ausente	Invoque al siguiente manejador con la función SEND-REQUEST
SEND-REQUEST	Solicitante de servicio; fase de solicitud	Presente (longitud = 0)	No aplicable	(error)
SEND-REQUEST	Solicitante de servicio; fase de solicitud	Ausente	Presente (longitud > 0)	Conmute a la fase de respuesta e invoque al manejador anterior con la función RECEIVE-RESPONSE
SEND-REQUEST	Solicitante de servicio; fase de solicitud	Ausente	Presente (longitud = 0)	(error)
SEND-REQUEST	Solicitante de servicio; fase de solicitud	Ausente	Ausente	Invoque al mismo manejador con la función NO-RESPONSE
RECEIVE-RESPONSE	Solicitante de servicio; fase de respuesta	No aplicable	Presente (longitud > 0)	Invoque al manejador anterior con la función RECEIVE-RESPONSE
RECEIVE-RESPONSE	Solicitante de servicio; fase de respuesta	No aplicable	Presente (longitud = 0)	(error)
RECEIVE-RESPONSE	Solicitante de servicio; fase de respuesta	No aplicable	Ausente	Invoque al mismo manejador con la función NO-RESPONSE
PROCESS-REQUEST	Proveedor de servicios; controlador de terminal	No aplicable	Presente (longitud > 0)	Invoque al manejador anterior con la función RECEIVE-RESPONSE
PROCESS-REQUEST	Proveedor de servicios; controlador de terminal	No aplicable	Presente (longitud = 0)	(error)
PROCESS-REQUEST	Proveedor de servicios; controlador de terminal	No aplicable	Ausente	Invoque al mismo manejador con la función NO-RESPONSE
HANDLER-ERROR	Proveedor o solicitante de servicio; cualquier fase	No aplicable	Presente (longitud > 0)	Invoque al manejador anterior con la función SEND-RESPONSE o la función RECEIVE-RESPONSE

DFHFUNCTION	Contexto	DFHREQUEST	DFHRESPONSE	Acción
HANDLER-ERROR	Proveedor o solicitante de servicio; cualquier fase	No aplicable	Presente (longitud = 0)	(error)
HANDLER-ERROR	Proveedor o solicitante de servicio; cualquier fase	No aplicable	Ausente	Invoque al mismo manejador con la función NO-RESPONSE

Suministro de manejadores de mensajes propios

Cuando desea realizar un proceso especializado en mensajes que fluyen entre un solicitante de servicio y un proveedor de servicios, y CICS no proporciona un manejador de mensajes que satisface sus necesidades, deberá proporcionarlo usted.

Acerca de esta tarea

En la mayoría de las situaciones, puede realizar todo el proceso que necesita con los manejadores de mensajes proporcionados por CICS. Por ejemplo, puede utilizar los manejadores de mensajes SOAP 1.1 y 1.2 que proporciona CICS para procesar mensajes SOAP. Pero hay ocasiones en las que deseará realizar sus propias operaciones especializadas en solicitudes y respuestas de servicio web. Para ello, debe proporcionar sus propios manejadores de mensajes.

Procedimiento

1. Escriba su propio programa de manejador de mensajes. Un manejador de mensajes es un programa CICS con una interfaz de canal. Puede escribir su programa en cualquiera de los lenguajes que soporta CICS, y utilizar cualquier mandato CICS en el subconjunto DPL dentro del programa.
2. Compile y edite el enlace del programa. Los programas de manejador de mensajes normalmente se ejecutan bajo la transacción CPIH, que se define con el atributo TASKDATALOC(ANY). Por lo tanto, cuando enlace-edite el programa, deberá especificar la opción AMODE(31).
3. Instale el programa en el sistema CICS de la forma habitual.
4. Defina el programa en el archivo de configuración de interconexión. Utilice el elemento <handler> para definir su manejador de mensajes. En el elemento <handler>, codifique un elemento <program> que contiene el nombre del programa.

Trabajar con mensajes en un manejador de mensajes que no es de terminal

Un manejador de mensajes que no es de terminal típico procesa un mensaje que pasa el control a otro manejador de mensajes en la interconexión.

Acerca de esta tarea

En un manejador de mensajes que no es de terminal, puede trabajar con una solicitud o respuesta, con o sin cambiarla, y pasarla al siguiente manejador de mensajes.

Nota: Aunque los servicios web normalmente utilizan mensajes SOAP que contienen XML, su manejador de mensajes funcionará también con otros formatos de mensaje

Procedimiento

1. Utilizando el contenido del contenedor DFHFUNCIÓN, determine si el mensaje pasado a este manejador de mensajes es una solicitud o una respuesta.

DFHFUNCIÓN	Solicitud o respuesta	Tipo de manejador de mensajes	De entrada o de salida
RECEIVE-REQUEST	Solicitud	No es de terminal	De entrada
SEND-RESPONSE	respuesta	No es de terminal	De salida
SEND-REQUEST	Solicitud	No es de terminal	De salida
RECEIVE-RESPONSE	respuesta	No es de terminal	De entrada

Consejo:

- Si DFHFUNCIÓN contiene PROCESS-REQUEST, el manejador de mensajes es un manejador de mensajes de terminal y estos pasos no se aplican.
 - Si DFHFUNCIÓN contiene HANDLER-ERROR, se está llamando al manejador para un proceso de error y estos pasos no se aplican.
2. Recupere la solicitud o respuesta desde el contenedor adecuado.
 - Si el mensaje es una solicitud, se pasa al programa en el contenedor DFHREQUEST. El contenedor DFHRESPONSE también está presente, con una longitud de cero.
 - Si el mensaje es una respuesta, se pasa al programa en el contenedor DFHRESPONSE.
 3. Lleve a cabo cualquier proceso del mensaje que sea necesario. En función de la finalidad del manejador de mensajes, puede:
 - Examinar el mensaje sin cambiarlo y pasarlo al siguiente manejador de mensajes en la interconexión.
 - Cambiar la solicitud y pasarla al siguiente manejador de mensajes en la interconexión.
 - Si el mensaje es una solicitud, puede ignorar los siguientes manejadores de mensajes en la interconexión y, en su lugar, construir un mensaje de respuesta.

Nota: Es el contenido de los contenedores que devuelve un manejador de mensajes el que determina qué manejador de mensajes se llama el siguiente.

Es un error si el manejador de mensajes no hace cambios a ninguno de los contenedores que se pasa.

Es un error para un programa de manejador de mensajes que devuelve lo siguiente:

- Un contenedor DFHRESPONSE vacío.
- Un contenedor DFHREQUEST que no está vacío y un contenedor DFHRESPONSE que no está vacío.
- Un contenedor DFHREQUEST vacío en la solicitud de salida.

Pasando un mensaje al siguiente manejador de mensajes en la interconexión:

En un manejador de mensajes que no es de terminal típico, procesará una solicitud o respuesta, cambiándola o no, y pasándola al siguiente manejador de mensajes.

Procedimiento

1. Devuelva el mensaje a la interconexión, modificado o no, en el contenedor adecuado.
 - Si el mensaje es una solicitud y lo ha cambiado, devuélvalo en el contenedor DFHREQUEST
 - Si el mensaje es una respuesta y lo ha cambiado, póngalo en el contenedor DFHRESPONSE
 - Si no ha cambiado el mensaje, ya está en el contenedor adecuado
2. Si el mensaje es una solicitud, suprima el contenedor DFHRESPONSE. Cuando se invoca un manejador de mensajes para una solicitud, los contenedores DFHREQUEST y DFHRESPONSE se pasan al programa; DFHRESPONSE tiene una longitud de cero. Sin embargo, es un error devolver DFHREQUEST y DFHRESPONSE.

Resultados

El mensaje se pasa al siguiente manejador de mensajes en la interconexión.

Forzar una transición a la fase de respuesta de la interconexión:

Cuando procesa una solicitud, hay veces que deberá generar una respuesta inmediata, en lugar de pasar la solicitud al siguientes manejador de mensajes en la interconexión.

Procedimiento

1. Suprimir contenedor DFHREQUEST.
2. Construya su respuesta y colóquela en el contenedor DFHRESPONSE.

Resultados

La respuesta se pasa al siguiente manejador de mensajes en la fase de respuesta de la interconexión.

Supresión de la respuesta:

En algunas situaciones, deseará asimilar una solicitud sin enviar una respuesta.

Procedimiento

1. Suprimir contenedor DFHREQUEST.
2. Suprimir contenedor DFHRESPONSE.

Gestión de mensajes de dirección única en una interconexión de solicitante de servicio:

Cuando una interconexión de solicitante de servicio envía una solicitud a un proveedor de servicios, normalmente, se espera que haya una respuesta y, que después de enviar la solicitud, los manejadores de mensajes de la interconexión se invocarán de nuevo cuando llega la respuesta. Algunos servicios web no envían una respuesta y, por lo tanto, debe intervenir para indicar que CICS no debe esperar una respuesta antes de invocar los manejadores de mensajes una segunda vez.

Acerca de esta tarea

Para hacer esto, asegúrese de que el contenedor DFHNORESPONSE está presente al final del proceso de interconexión en la fase de solicitud. Generalmente, esto lo hace el código de nivel de aplicación, porque el saber si se espera una respuesta está en la aplicación:

- Para las aplicaciones desplegadas con el asistente de servicios web CICS, el código de CICS creará el contenedor.
- Las aplicaciones que no están desplegadas con el asistente crearán el contenedor antes de invocar la aplicación.

Si crea o destruye el contenedor DFHNORESPONSE en un manejador de mensajes, debe asegurarse de que haciéndolo no alterará el protocolo de mensajes entre el solicitante de servicio y el proveedor.

Trabajar con mensajes en un manejador de mensajes de terminal

Un controlador de terminal típico procesa una solicitud, invoca un programa de aplicación y genera una respuesta.

Acerca de esta tarea

Nota: Aunque los servicios web normalmente utilizan mensajes SOAP que contienen XML, su manejador de mensajes funcionará también con otros formatos de mensaje

En un manejador de mensajes de terminal, puede trabajar con una solicitud, y, opcionalmente, generar una respuesta y devolverla junto con la interconexión. Un controlador de terminal típico utilizará la solicitud como entrada para un programa de aplicación y utilizará la respuesta del programa de aplicación para construir la respuesta.

Procedimiento

1. Utilizando el contenido del contenedor DFHFUNCTION, determine que el mensaje pasado a este manejador es una solicitud, y que el manejador se está invocando como un controlador de terminal.

DFHFUNCTION	Solicitud o respuesta	Tipo de manejador	De entrada o de salida
PROCESS-REQUEST	Solicitud	Terminal	De entrada

Consejo:

- Si DFHFUNCTION contiene cualquier otro valor, el manejador no es un controlador de terminal y estos pasos no se aplican.
2. Recupere la solicitud del contenedor DFHREQUEST. El contenedor DFHRESPONSE también está presente, con una longitud de cero.
 3. Lleve a cabo cualquier proceso del mensaje que sea necesario. Normalmente, un controlador de terminal invocará un programa de aplicación.
 4. Construya su respuesta y colóquela en el contenedor DFHRESPONSE. Si no hay respuesta, debe suprimir el contenedor DFHRESPONSE.

Resultados

La respuesta se pasa al siguiente manejador en la fase de respuesta de la interconexión. El manejador se invoca para la función SEND-RESPONSE. Si no hay

respuesta, se invoca el siguiente manejador para la función NO-RESPONSE.

Manejo de errores

Los manejadores de mensajes deben estar diseñados para manejar errores que pueden producirse en la interconexión.

Acerca de esta tarea

Cuando se produce un error en un programa de manejador de mensajes, el programa se invoca de nuevo para el proceso de errores. El proceso de errores siempre tiene lugar en la fase de respuesta de la interconexión; si el error se produce en la fase de solicitud, se ignoran los manejadores posteriores en la fase de solicitud.

Por lo tanto, en la mayoría de los casos, debe escribir el programa de manejador para gestionar los errores que puedan producirse.

Procedimiento

1. Compruebe que el contenedor DFHFUNCTIOn contiene HANDLER-ERROR, que indica que se ha llamado al manejador de mensajes para el proceso de errores.

Consejo:

- Si DFHFUNCTIOn contiene cualquier otro valor, el manejador de mensajes no se ha invocado para el proceso de errores y estos pasos no se aplican.
2. Analice la información de error y determine si el manejador de errores puede recuperarse del error construyendo una respuesta adecuada.
El contenedor DFHERROR contiene información sobre el error. Para obtener información detallada sobre este contenedor, consulte “Contenedor DFHERROR” en la página 152.
El contenedor DFHRESPONSE también está presente, con una longitud de cero.
 3. Lleve a cabo cualquier proceso de recuperación.
 - Si el manejador de mensajes puede recuperarse, construya una respuesta y devuélvala en el contenedor DFHRESPONSE.
 - Si el manejador de mensajes puede recuperarse, pero no se necesita ninguna respuesta, suprima el contenedor DFHRESPONSE y, en su lugar, devuelva el contenedor DFHNORESPONSE.
 - Si el manejador de mensajes no se puede recuperar, devuelva el contenedor DFHRESPONSE sin cambiar (es decir, con una longitud de cero).

Resultados

Si el manejador de mensajes puede recuperarse del error, el proceso de interconexión continúa normalmente. Si no es así, CICS genera un error SOAP que contiene información sobre el error. En caso de una terminación anómala de la transacción, el código de terminación anómala se incluye en el error.

Interfaz de manejadores de mensajes

La interconexión de CICS se enlaza con los manejadores de mensajes utilizando un canal que contiene varios contenedores. Algunos contenedores son opcionales, otros son necesarios para todos los manejadores de mensajes y otros los utilizan algunos manejadores de mensajes pero no otros.

Antes de invocar un manejador, algunos o todos los contenedores se llenan con información que puede utilizar el manejador para realizar su trabajo. Los contenedores devueltos por el manejador determinan el proceso posterior y se pasan a manejadores posteriores en la interconexión.

Manejadores de mensajes SOAP

Los manejadores de mensajes SOAP son manejadores de mensajes suministrados por CICS que puede incluir en su interconexión para procesar mensajes SOAP 1.1 y SOAP. Puede utilizar los manejadores de mensajes SOAP en una interconexión de solicitante de servicio o de proveedor de servicios.

En la entrada, los manejadores de mensajes SOAP analizan mensajes SOAP de entrada y extraen el elemento <Body> de SOAP para su uso por el programa de aplicación. En la salida, los manejadores construyen el mensaje SOAP completo, utilizando el elemento <Body> que proporciona la aplicación.

Si utiliza cabeceras SOAP en los mensajes, los manejadores SOAP pueden invocar programas de proceso de cabeceras escritos por el usuario que le permiten procesar las cabeceras en los mensajes de entrada y añadirlos a mensajes de salida.

Los manejadores de mensajes SOAP, y cualquier programa de proceso de cabeceras, se especifican en el archivo de configuración de interconexión. Para interconexiones que no soportan Java, se deben especificar los manejadores de mensajes <cics_soap_1.1_handler> o <cics_soap_1.2_handler>. Para interconexiones que soportan Java, se deben especificar los manejadores de mensajes <cics_soap_1.1_handler_java>, o <cics_soap_1.2_handler_java>.

Generalmente, necesitaremos sólo un manejador SOAP en una interconexión. Sin embargo, hay otras situaciones en las que se necesitan más de una. Por ejemplo, puede asegurarse de que las cabeceras SOAP se procesan en una secuencia particular definiendo varios manejadores SOAP.

No debe definir los manejadores de mensajes <cics_soap_1.1_handler> y <cics_soap_1.2_handler> o los manejadores de mensajes <cics_soap_1.1_handler_java> y <cics_soap_1.2_handler_java> en la misma interconexión. Si desea que la interconexión procese los mensajes SOAP 1.1 y SOAP 1.2, debe utilizar el manejador de mensajes <cics_soap_1.2_handler> o <cics_soap_1.2_handler_java>.

Programas de proceso de cabeceras

Los programas de proceso de cabeceras son programas CICS escritos por el usuario que están enlazados desde los manejadores de mensajes SOAP 1.1 y SOAP 1.2 proporcionados por CICS, para procesar bloques de cabecera SOAP.

Puede escribir el programa de proceso de cabeceras en cualquiera de los lenguajes que soporta CICS, y utilizar cualquier mandato CICS en el subconjunto DPL. El programa de proceso de cabeceras puede enlazar con otros programas CICS.

Los programas de proceso de cabeceras tienen una interfaz de canal; los contenedores tienen información que el programa de cabeceras puede examinar o modificar, incluido el bloque de cabecera SOAP para el que se invoca el programa, y el cuerpo de mensaje SOAP.

El canal y los contenedores que puede utilizar el programa de proceso de cabeceras se describen en “Interfaz del programa de proceso de cabeceras” en la página 147.

Otros contenedores tienen información sobre el entorno en el que se invoca el programa de cabeceras, por ejemplo:

- El ID de transacción bajo el que se ha invocado el programa de cabeceras
- Si el programa se ha invocado para una interconexión de solicitante o proveedor de servicios
- Si el mensaje que se está procesando es una solicitud o una respuesta

Los programas de proceso de cabeceras normalmente se ejecutan bajo el CPIH de transacción que se define con el atributo TASKDATALOC(ANY). Por lo tanto, cuando enlace-edite el programa, deberá especificar la opción AMODE(31).

Cómo se invocan los programas de proceso de cabeceras para una solicitud SOAP

Los elementos <cics_soap_1.1_handler>, <cics_soap_1.2_handler>, <cics_soap_1.1_handler_java> y <cics_soap_1.2_handler_java> en una configuración de interconexión contienen cero, uno o más elementos <headerprogram>, cada uno de los cuales contiene los siguientes hijos:

```
<program_name>
<namespace>
<localname>
<mandatory>
```

Cuando la interconexión se procesa en un mensaje SOAP de entrada (una solicitud en el caso de un proveedor de servicios, una respuesta en el caso de un solicitante de servicio), el programa de cabeceras especificado en el elemento <program_name> se invoca o no, dependiendo de los siguientes elementos:

- El contenido de los elementos <namespace>, <localname> y <mandatory>
- El valor de determinados atributos del elemento raíz de la cabecera SOAP misma (el atributo **actor** para SOAP 1.1; el atributo **rol** para SOAP 1.2)

Las reglas siguientes determinan si el programa de cabeceras se invocará en un caso determinado:

El elemento <mandatory> en el archivo de configuración de interconexión

Si el elemento contiene true (o 1), el programa de proceso de cabeceras se invoca al menos una vez, aunque ninguna de las cabeceras del mensaje SOAP se haya seleccionado para procesarse por las reglas siguientes:

- Si no se ha seleccionado ninguno de los bloques de cabecera, el programa de proceso de cabeceras se invoca una vez.
- Si alguno de los bloques de cabecera está seleccionado por alguna de las reglas restantes, el programa de proceso de cabeceras se invoca una vez para cada cabecera seleccionada.

Atributos en el bloque de cabecera SOAP

Para SOAP 1.1, un bloque de cabecera es elegible para su proceso sólo si el atributo **actor** está ausente, o tiene un valor de `http://schemas.xmlsoap.org/soap/actor/next`

Para SOAP 1.2, un bloque de cabecera es elegible para su proceso sólo si el atributo **role** está ausente, o tiene uno de los valores siguientes:

```
http://www.w3.org/2003/05/soap-envelope/role/next
http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver
```

Un bloque de cabecera que es elegible para procesar no se procesa a menos que esté seleccionado por la regla siguiente.

Los elementos <namespace> y <localname> en el archivo de configuración de interconexión

Un bloque de cabecera que es elegible para el proceso según la regla anterior se selecciona para su proceso sólo si se cumplen las siguientes condiciones:

- El nombre del elemento raíz del bloque de cabecera coincide con el elemento <localname> en el archivo de configuración de la interconexión
- El espacio de nombres del elemento raíz coincide con el elemento <namespace> en el archivo de configuración de interconexión

Por ejemplo, tenga en cuenta este bloque de cabecera:

```
<t:myheaderblock xmlns:t="http://mynamespace" ...> .... </t:myheaderblock>
```

Sujeto a las demás reglas, el bloque de cabecera está seleccionado para procesarse cuando las líneas siguientes se codifican en el archivo de configuración de interconexión:

```
<namespace>http://mynamespace</namespace>  
<localname>myheaderblock</localname>
```

Los elementos <localname> pueden contener un * para indicar que todos los bloques de cabecera del espacio de nombres se deben procesar. Por lo tanto, se selecciona el mismo bloque de cabecera por el siguiente código:

```
<namespace>http://mynamespace</namespace>  
<localname>*</localname>
```

Cuando el mensaje SOAP contiene más de una cabecera, el programa de proceso de cabeceras se invoca una vez para cada cabecera coincidente, pero la secuencia en la que se procesan las cabeceras no está definida.

Los manejadores de mensajes SOAP proporcionados por CICS seleccionan los programas de proceso de cabeceras que se invocan en base a los bloques de cabecera presentes en el mensaje SOAP en el momento en que el manejador de mensajes lo recibe. Por lo tanto, un programa de proceso de cabeceras nunca se invoca como resultado de que un bloque de cabecera se añade a un mensaje en el mismo manejador de mensajes SOAP. Si desea procesar la nueva cabecera (o cualquier cabecera modificada) en la interconexión, debe definir otro manejador de mensajes SOAP en la interconexión.

Para un mensaje de salida (una solicitud en un solicitante de servicio, una respuesta en un proveedor de servicios), los manejadores de mensajes SOAP proporcionados por CICS crean un mensaje SOAP que no contiene ninguna cabecera. Para añadir una o más cabeceras al mensaje, debe escribir un programa de manejador de cabecera para añadirlo a las cabeceras. Para asegurarse de que este manejador de cabeceras se invoca, debe definirlo en el archivo de configuración de interconexión y especificar <mandatory>true</mandatory>.

Si el manejador de cabecera se invoca en la fase de solicitud de una interconexión, se invoca de nuevo en la fase de respuesta, aunque el mensaje que fluye en la fase de respuesta no contenga una cabecera coincidente.

Interfaz del programa de proceso de cabeceras

Los manejadores de mensajes SOAP 1.1 y SOAP 1.2 proporcionados por CICS enlazan con los programas de proceso de cabeceras utilizando el canal DFHHHC-V1. Los contenedores que se pasan al canal incluyen varios

contenedores que son específicos para la interfaz del programa de proceso de cabeceras, y conjuntos de *contenedores de contexto* y *contenedores de usuarios* accesibles para todos los programas de proceso de cabeceras y programas de manejadores de mensajes en la interconexión.

El contenedor DFHHEADER es específico para la interfaz del programa de proceso de cabeceras. Otros contenedores están disponibles en otras partes de la interconexión, pero tienen usos específicos en un programa de proceso de cabeceras. Los contenedores de esta categoría son DFHWS-XMLNS, DFHWS-BODY y DFHXMLSS-PARSE.

Nota: Aunque el servicio web que utiliza Axis2 para procesar mensajes SOAP puede utilizar la interfaz del programa de proceso de cabeceras, es más eficaz escribir sus propios manejadores Axis2 en Java para procesar las cabeceras SOAP. Para obtener más información sobre la creación de manejadores Axis2, consulte Escribir su propio módulo Axis2

Contenedor DFHHEADER

Cuando se invoca el programa de proceso de cabeceras, DFHHEADER contiene el único bloque de cabecera que ha provocado que el programa de proceso de cabeceras se realice. Cuando se especifica el programa de cabecera con `<mandatory>true</mandatory>` o `<mandatory>1</mandatory>` en el archivo de configuración de interconexión, se invoca incluso cuando no hay bloque de cabecera coincidente en el mensaje SOAP. En este caso, el contenedor DFHHEADER tiene una longitud de cero. Este es el caso cuando se invoca un proceso de cabeceras para añadir un bloque de cabecera a un mensaje SOAP que no tiene bloques de cabecera.

El mensaje SOAP que crea CICS no tiene cabeceras inicialmente. Si desea añadir cabeceras al mensaje, debe asegurarse de que se invoca al menos un programa de proceso de cabeceras, especificando `<mandatory>true</mandatory>` o `<mandatory>1</mandatory>`.

Cuando el programa de cabeceras regresa, el contenedor DFHHEADER debe contener cero o más bloques de cabecera que CICS inserta en el mensaje SOAP en lugar del original:

- Puede devolver el bloque de cabecera original sin modificar.
- Puede modificar el contenido del bloque de cabecera.
- Puede añadir uno o más bloques de cabecera nuevos al bloque original.
- Puede sustituir el bloque de cabecera original por uno o varios bloques diferentes.
- Puede suprimir el bloque de cabecera completamente.

Contenedor DFHWS-XMLNS

Cuando se invoca el programa de proceso de cabeceras, DFHWS-XMLNS contiene información sobre los espacios de nombres XML que se declaran en el sobre SOAP. El programa de cabeceras puede utilizar esta información para realizar las siguientes tareas:

- Resolver nombres calificados que encuentra en el bloque de cabecera
- Construir nombres calificados en bloques de cabecera nuevos o modificados.

La información del espacio de nombres consta de una lista de declaraciones de espacio de nombres, que utiliza la notación XML estándar para declarar espacios de nombres. Las declaraciones de espacio de nombres en DFHWS-XMLNS están separadas por espacios. Por ejemplo:

```
xmlns:na='http://abc.example.org/schema' xmlns:nx='http://xyz.example.org/schema'
```

Puede añadir más declaraciones de espacio de nombres al sobre SOAP añadiéndolas al contenido de DFHWS-XMLNS. Sin embargo, los espacios de nombres cuyo alcance es un bloque de cabecera SOAP o un cuerpo SOAP se declaran mejor en el bloque de cabecera o el cuerpo respectivamente. Se recomienda no suprimir declaraciones de espacio de nombres desde el contenedor DFHWS-XMLNS en un programa de proceso de cabeceras porque los elementos XML que no están visibles en el programa pueden basarse en ellas.

Contenedor DFHWS-BODY

Este contenedor contiene la sección de cuerpo del sobre SOAP. El programa de proceso de cabeceras puede modificar el contenido.

Cuando se invoca el programa de proceso de cabeceras, DFHWS-BODY contiene el elemento SOAP <Body>.

Cuando regresa el programa de cabeceras, el contenedor DFHWS-BODY debe contener de nuevo un <Body> SOAP válido, que CICS inserta en el mensaje SOAP en lugar del original:

- Puede devolver el cuerpo original sin modificar.
- Puede modificar el contenido del cuerpo.

No debe suprimir el cuerpo SOAP por completo, ya que todos los mensajes SOAP deben contener un elemento <Body>.

Contenedor DFHXMLSS-PARSE

Cuando utiliza los elementos <cics_soap_1.1_handler> o <cics_soap_1.2_handler> en la configuración de interconexión, y se invoca el programa de cabeceras, DFHXMLSS-PARSE contiene los registros de XML System Services (XMLSS) para esa cabecera. Este contenedor no se crea cuando se utilizan los elementos <cics_soap_1.1_handler_java> o <cics_soap_1.2_handler_java>.

Control, contexto y contenedores de usuario

Siempre que se describen los contenedores, la interfaz pasa los *contenedores de control*, *contenedores de contexto*, y *contenedores de usuarios* al canal DFHHHC-V1.

Para obtener más información acerca de estos contenedores, consulte “Contenedores utilizados en la interconexión” en la página 150.

Direccionamiento dinámico de solicitudes de entrada en un controlador de terminal

Cuando el controlador de terminal de una interconexión de proveedor de servicios es uno de los manejadores SOAP proporcionados por CICS, el programa de manejador de aplicación de destino especificado en el contenedor **DFHWS-APPHANDLER** es, en algunos casos, elegible para el direccionamiento dinámico. Todo el proceso de interconexión anterior al programa de manejador de aplicación se lleva a cabo siempre localmente en la región de CICS que ha recibido el mensaje SOAP.

La transacción que ejecuta el programa de manejador de aplicación de destino es elegible para el direccionamiento cuando una de las condiciones siguientes es verdadera:

- La transacción bajo la que la interconexión está procesando el mensaje se define como DYNAMIC o REMOTE. Esta transacción se define en la URIMAP que se utiliza para correlacionar el URI desde el mensaje SOAP de entrada.
- Un programa en la interconexión ha cambiado el contenido del contenedor **DFHWS-USERID** de su valor inicial.
- Un programa en la interconexión ha cambiado el contenido del contenedor **DFHWS-TRANID** de su valor inicial.
- Existe una cabecera WS-AT SOAP en el mensaje SOAP de entrada.

En todos los escenarios anteriores, se produce una conmutación de tareas durante el proceso de interconexión. La segunda tarea se ejecuta bajo la transacción especificada en el contenedor **DFHWS-TRANID**. Esta conmutación de tareas proporciona una oportunidad para que tenga lugar el direccionamiento dinámico, pero sólo si se utiliza MRO para conectar las regiones de CICS. Además, la región de CICS a la que está direccionando debe soportar canales y contenedores.

El direccionamiento sólo tiene lugar si la definición TRANSACTION para la transacción con nombre en **DFHWS-TRANID** especifica uno de los siguientes conjuntos de atributos:

DYNAMIC(YES)

La transacción se direcciona utilizando el modelo de direccionamiento distribuido, en el que se especifica el programa de direccionamiento en el parámetro de inicialización del sistema **DSRTPGM**.

DYNAMIC(NO) REMOTESYSTEM(sysid)

La transacción se direcciona al sistema identificado por *sysid*.

Para obtener más información sobre el direccionamiento de solicitudes de servicio web, consulte la nota técnica: *Direccionamiento de los servicios web de CICS de la modalidad de proveedor*.

Para aplicaciones desplegadas con el asistente de servicios web de CICS, hay una segunda oportunidad de direccionar dinámicamente la solicitud, en el punto donde CICS enlaza con el programa de usuarios. La solicitud se direcciona utilizando el modelo de direccionamiento dinámico, en el que se especifica el programa de direccionamiento en el parámetro de inicialización del sistema **DTRPGM**. La elegibilidad para el direccionamiento se determina, en este caso, por las características del programa. Si está utilizando un canal y contenedores cuando enlaza con el programa, sólo puede direccionar dinámicamente la solicitud a regiones de CICS que están en V3.1 o superior. Si utilizan un COMMAREA, esta restricción no se aplica.

Cuando una solicitud se ha direccionado dinámicamente a una región de destino, no se puede direccionar dinámicamente desde el destino a una tercera región, aunque la transacción se defina como ROUTABLE(YES) y DYNAMIC(YES). Sin embargo, la transacción puede direccionarse de forma estática desde la región de destino a una tercera región.

Contenedores utilizados en la interconexión

Una interconexión suele constar de varios programas manejadores de mensajes y, cuando se utilizan los manejadores de mensajes SOAP suministrados por CICS,

varios programas de proceso de cabeceras. CICS utiliza contenedores para pasar información a y de estos programas. Los programas también utilizan contenedores para comunicarse con otros programas de la interconexión.

La interconexión de CICS enlaza con los manejadores de mensajes y a los programas de proceso de cabeceras utilizando un canal que tiene varios contenedores. Algunos contenedores son opcionales, otros los necesitan todos los manejadores de mensajes y otros los utilizan unos manejadores de mensajes pero no otros.

Antes de invocar un manejador, algunos o todos los contenedores se llenan con información que puede utilizar el manejador para realizar su trabajo. Los contenedores devueltos por el manejador determinan el proceso posterior y se pasan a manejadores posteriores en la interconexión.

Los contenedores se pueden categorizar de estas formas:

Contenedores de control

Estos contenedores son fundamentales para el funcionamiento de la interconexión. Los manejadores pueden utilizar los contenedores de control para modificar la secuencia en la que se procesan los manejadores. CICS define los nombres de los contenedores de control y comienzan por los caracteres DFH.

Contenedores de contexto

Estos contenedores contienen información sobre el entorno en el que se invocan los manejadores. CICS pone información en estos contenedores antes de invocar el primer manejador de mensajes, pero, en algunos casos, los manejadores son libres para cambiar el contenido o suprimir el contenedor. Los cambios en los contenedores de contexto no afectan directamente a la secuencia en la que se invocan los manejadores. CICS define los nombres de los contenedores de contexto y comienzan por los caracteres DFH.

Contenedores del programa de proceso de cabeceras

Estos contenedores contienen información que utilizan los programas de proceso de cabeceras que se invocan desde los manejadores de mensajes SOAP proporcionados por CICS. Para obtener información sobre estos contenedores, consulte Interfaz del programa de proceso de cabeceras.

Contenedores de seguridad

Estos contenedores contienen información que utiliza la interfaz de cliente de confianza y el manejador de mensajes de seguridad para procesar señales de seguridad utilizando un Security Token Service (STS). CICS define los nombres de los contenedores de seguridad y comienzan por los caracteres DFH.

Contenedores generados

Estos contenedores tienen datos de un mensaje SOAP, como matrices de variables y cadenas largas, que se pasan a y desde el programa de aplicación para su proceso. CICS crea automáticamente estos contenedores durante el proceso de interconexión, y los nombres que empiezan por los caracteres DFH.

Contenedores de usuarios

Estos contenedores contienen la información que un manejador de mensajes necesita pasar a otro. El uso de los contenedores de usuarios es

un asunto de los manejadores de mensajes. Puede elegir sus propios nombres para estos contenedores, pero no debe utilizar nombres que empiezan por DFH.

Contenedores de control

Los contenedores de control son fundamentales para el funcionamiento de la interconexión. Los manejadores pueden utilizar los contenedores de control para modificar la secuencia en la que se procesan los manejadores.

Contenedor DFHERROR:

DFHERROR es un contenedor de DATATYPE(BIT) que se utiliza para transmitir información sobre los errores de interconexión a otros manejadores de mensajes.

Tabla 5. Estructura del contenedor DFHERROR.

Nombre de campo	Longitud (bytes)	Contenido
PIISNEB-MAJOR-VERSION	1	"1"
PIISNEB-MINOR-VERSION	1	"1"
PIISNEB-ERROR-TYPE	1	Un valor numérico que indica el tipo de error. Los valores se describen en Tabla 6.
PIISNEB-ERROR-MODE	1	<p>P El error se ha producido en una interconexión de proveedor</p> <p>R El error se ha producido en una interconexión de solicitante</p> <p>T El error se ha producido en un cliente de confianza</p>
PIISNEB-ABCODE	4	El código de terminación anómala cuando el error se asocia a una terminación anómala de transacción.
PIISNEB-ERROR-CONTAINER1	16	El nombre del contenedor cuando el error se asocia a un contenedor.
PIISNEB-ERROR-CONTAINER2	16	El nombre del segundo contenedor cuando el error se asocia a más de un contenedor.
PIISNEB-ERROR-NODE	8	El nombre del programa de manejador en el que se ha producido el error.

Tabla 6. Valores para el campo PIISNEB-ERROR-TYPE

Valor de PIISNEB-ERROR-TYPE	Significado
1	El programa de manejador ha fallado. El código de terminación anómala está en el campo PIISNEB-ABCODE.

Tabla 6. Valores para el campo PIISNEB-ERROR-TYPE (continuación)

Valor de PIISNEB-ERROR-TYPE	Significado
2	El contenedor que necesitaba el manejador estaba vacío. El nombre del contenedor está en el campo PIISNEB-ERROR-CONTAINER1.
3	Faltaba el contenedor que necesitaba el manejador. El nombre del contenedor está en el campo PIISNEB-ERROR-CONTAINER1.
4	Se han pasado dos contenedores al manejador cuando sólo se esperaba uno. Los nombres de los contenedores están en los campos PIISNEB-ERROR-CONTAINER1 y PIISNEB-ERROR-CONTAINER2.
5	Ha fallado el intento de enlazar con el programa destino. Si el programa destino ha fallado, el código de terminación anómala está en el contenedor PIISNEB-ABCODE.
6	El gestor de interconexión no ha podido comunicarse con un servidor remoto debido a un error en el transporte subyacente.
7	El contenedor DFHWS-STSACTION tiene un error. Falta, está dañado o contiene un valor incorrecto.
8	DFHPIRT no ha podido iniciar la interconexión.
9	DFHPIRT no ha podido colocar un mensaje en un contenedor.
10	DFHPIRT no ha podido obtener un mensaje de un contenedor.
11	Se ha producido un error no gestionado.

La declaración COBOL de la estructura del contenedor es ésta:

```

01 PIISNEB.
  02 PIISNEB-MAJOR-VERSION PIC X(1).
  02 PIISNEB-MINOR-VERSION PIC X(1).
  02 PIISNEB-ERROR-TYPE PIC X(1).
  02 PIISNEB-ERROR-MODE PIC X(1).
  02 PIISNEB-ABCODE PIC X(4).
  02 PIISNEB-ERROR-CONTAINER1 PIC X(16).
  02 PIISNEB-ERROR-CONTAINER2 PIC X(16).
  02 PIISNEB-ERROR-NODE PIC X(8).
  
```

La tabla siguiente muestra los libros de copias de lenguaje que correlacionan el contenedor.

Tabla 7. Libros de copias que correlacionan el contenedor

Idioma	Libro de copias
COBOL	DFHPIUCO
PL/I	DFHPIUCL
C y C++	dfhpiuch.h
Ensamblador	DFHPIUCD

Contenedor DFHFUNCTION:

DFHFUNCTION es un contenedor de DATATYPE(CHAR) que contiene una serie de 16 caracteres que indica dónde se está invocando un programa en una interconexión.

La serie tiene uno de los siguientes valores. Las posiciones de carácter más hacia la derecha se rellenan con caracteres en blanco.

RECEIVE-REQUEST

El manejador es un controlador que no es de terminal en una interconexión de proveedor de servicios, y se está invocando para procesar un mensaje de solicitud de entrada. En la entrada para el manejador, el mensaje está en el contenedor de control DFHREQUEST.

SEND-RESPONSE

El manejador es un manejador que no es de terminal en una interconexión de proveedor de servicios, y se está invocando para procesar un mensaje de respuesta de salida. En la entrada para el manejador, el mensaje está en el contenedor de control DFHRESPONSE.

SEND-REQUEST

El manejador se está invocando a través de una interconexión que está enviando una solicitud; es decir, en un solicitante de servicio que está procesando un mensaje de salida

RECEIVE-RESPONSE

El manejador se está invocando a través de una interconexión que está recibiendo una respuesta; es decir, en un solicitante de servicio que está procesando un mensaje de entrada

PROCESS-REQUEST

El manejador se está invocando como manejador de terminal de una interconexión de proveedor de servicios

NO-RESPONSE

El manejador se está invocando después de procesar una solicitud, cuando no se ha procesado una respuesta.

HANDLER-ERROR

El manejador se está invocando porque se ha detectado un error.

En una interconexión de proveedor de servicio que procesa una solicitud y devuelve una respuesta, los valores de DFHFUNCTION que se dan son RECEIVE-REQUEST, PROCESS-REQUEST y SEND-RESPONSE. Figura 22 en la página 155 muestra la secuencia en la que se invocan los manejadores y los valores de DFHFUNCTION que se pasan a cada manejador.

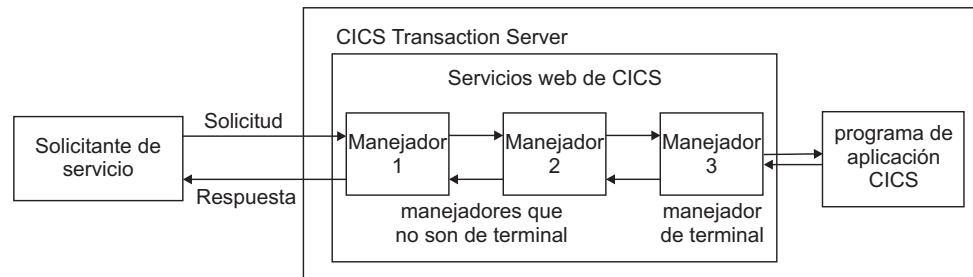


Figura 22. Secuencia de manejadores en una interconexión de proveedor de servicios

Secuencia	Manejador	DFHFUNCTION
1	Manejador 1	RECEIVE-REQUEST
2	Manejador 2	RECEIVE-REQUEST
3	Manejador 3	PROCESS-REQUEST
4	Manejador 2	SEND-RESPONSE
5	Manejador 1	SEND-RESPONSE

En una interconexión de solicitante de servicio que envía una solicitud y recibe una respuesta, los valores de DFHFUNCTION que se dan son SEND-REQUEST y RECEIVE-RESPONSE. Figura 23 muestra la secuencia en la que se invocan los manejadores y los valores de DFHFUNCTION que se pasan a cada manejador.

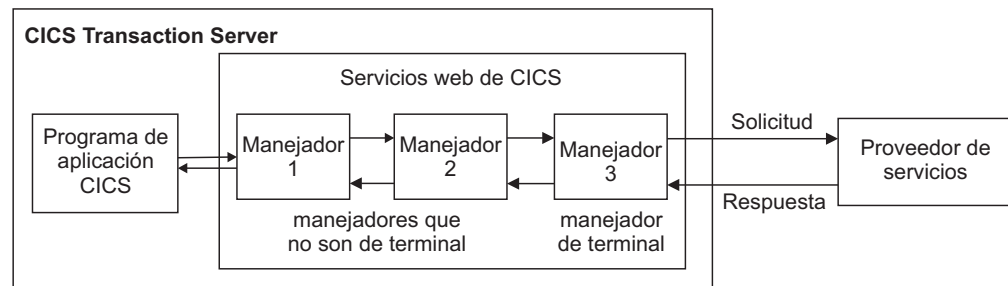


Figura 23. Secuencia de manejadores en una interconexión de solicitante de servicio

Secuencia	Manejador	DFHFUNCTION
1	Manejador 1	SEND-REQUEST
2	Manejador 2	SEND-REQUEST
3	Manejador 3	SEND-REQUEST
4	Manejador 3	RECEIVE-RESPONSE
5	Manejador 2	RECEIVE-RESPONSE
6	Manejador 1	RECEIVE-RESPONSE

Los valores de DFHFUNCTION que se pueden encontrar en un manejador de mensajes dado depende de si la interconexión es un proveedor o un solicitante, si

la interconexión está en la fase de solicitud o respuesta y si el manejador es un controlador de terminal o no es de terminal. La tabla siguiente resume cuando se produce cada valor:

Valor de DFHFUNCT	Interconexión de solicitante o proveedor	Fase de interconexión	Controlador de terminal o no terminal
RECEIVE-REQUEST	Proveedor	Fase de solicitud	No de terminal
SEND-RESPONSE	Proveedor	Fase de respuesta	No de terminal
SEND-REQUEST	Solicitante	Fase de solicitud	No de terminal
RECEIVE-RESPONSE	Solicitante	Fase de respuesta	No de terminal
PROCESS-REQUEST	Proveedor	Fase de solicitud	Terminal
NO-RESPONSE	Ambos	Fase de respuesta	No de terminal
HANDLER-ERROR	Ambos	Ambos	Ambos

Contenedor DFHHTTPMETHOD:

Se trata de un contenedor DATATYPE(CHAR) que se encuentra disponible para los programas de aplicación en todas las interconexiones CICS del modo de proveedor HTTP.

Este contenedor tiene 8 caracteres de largo y conserva el nombre del método HTTP utilizado en la solicitud entrante. No se rellena si la solicitud no ha llegado a través de HTTP.

Contenedor DFHHTTPSTATUS:

DFHHTTPSTATUS es un contenedor de DATATYPE(CHAR) que se utiliza para especificar el texto de estado y el código de estado HTTP para un mensaje producido en la fase de respuesta de una interconexión de proveedor de servicios.

El contenido del contenedor DFHHTTPSTATUS debe ser el mismo que la línea de estado inicial de un mensaje de respuesta HTTP que tiene la siguiente estructura:

HTTP/1.1 nnn tttttttt

HTTP/1.1

La versión y release de HTTP.

nnn El código de estado de HTTP decimal de 3 dígitos que se va a devolver.

tttttttt

El texto de estado legible asociado al código de estado nnn.

La serie siguiente es un ejemplo del contenido:

HTTP/1.1 412 Precondition Failed

El contenedor DFHHTTPSTATUS se ignora cuando la interconexión utiliza el transporte WebSphere MQ.

Si el contenedor contiene más de 45 bytes de datos, CICS envía 45 bytes e ignora el resto de los datos.

Contenedor DFHMEDIATYPE:

DFHMEDIATYPE es un contenedor de DATATYPE(CHAR) que se utiliza para especificar el tipo de soporte para un mensaje producido en la fase de respuesta de una interconexión de proveedor de servicios.

El contenido del contenedor DFHMEDIATYPE debe constar de un tipo y un subtipo separados por un carácter de barra inclinada. Las series siguientes muestran dos ejemplos de contenido correcto para el contenedor DFHMEDIATYPE:

```
text/plain  
image/svg+xml
```

El contenedor DFHMEDIATYPE se ignora cuando la interconexión utiliza el transporte WebSphere MQ.

Contenedor DFHNORESPONSE:

DFHNORESPONSE es un contenedor de DATATYPE(CHAR) que, en la fase de solicitud de una interconexión de solicitante de servicio, indica que no se espera que el proveedor de servicios devuelva una respuesta.

El contenido del contenedor DFHNORESPONSE no está definido; los manejadores de mensajes que necesitan saber si se espera que el proveedor de servicios devuelva una respuesta sólo necesitan determinar si el contenedor está presente o no:

- Si el contenedor DFHNORESPONSE está presente, no se espera una respuesta.
- Si el contenedor no está presente, *se* espera una respuesta.

Inicialmente, esta información la proporciona la aplicación de solicitante de servicio, basada en el protocolo utilizado con el proveedor de servicios. Por lo tanto, se aconseja no suprimir este contenedor en un manejador de mensajes (o crearlo, si no existe), porque haciéndolo puede alterar el protocolo entre los puntos finales.

A diferencia de la fase de solicitud de una interconexión de solicitante de servicio, el uso de este contenedor no está definido.

Contenedor DFHREQUEST:

DFHREQUEST es un contenedor de DATATYPE(CHAR) que contiene el mensaje de solicitud que se procesa en la fase de solicitud de una interconexión.

Si se configura uno de los manejadores de mensajes SOAP proporcionados por CICS en la interconexión, el contenedor DFHREQUEST se actualiza para incluir las cabeceras de mensajes SOAP en el sobre SOAP. Si el mensaje se construyó mediante el manejador de mensajes SOAP proporcionado por CICS, y no se ha modificado posteriormente, DFHREQUEST contiene un sobre SOAP completo y todo su contenido está en la página de códigos UTF-8.

El contenedor DFHREQUEST está presente en la solicitud cuando se llama a un manejador de mensajes, y el contenedor DFHFUNCTON contiene RECEIVE-REQUEST o SEND-REQUEST.

En esta situación, el protocolo normal es devolver DFHREQUEST a la interconexión con el mismo contenido o con contenido modificado. El proceso de

la fase de solicitud de interconexión continúa normalmente, con el siguiente programa de manejador de mensajes en la interconexión, si hubiera uno.

Como alternativa, el manejador de mensajes puede suprimir el contenedor DFHREQUEST, y poner una respuesta en el contenedor DFHRESPONSE. De esta forma, la secuencia normal del proceso se invierte y el proceso continúa con la fase de respuesta de la interconexión.

Contenedor DFHRESPONSE:

DFHRESPONSE es un contenedor de DATATYPE(CHAR) que contiene el mensaje de respuesta que se procesa en la fase de respuesta de una interconexión. Si el mensaje se construyó mediante el manejador de mensajes SOAP proporcionado por CICS, y no se ha modificado posteriormente, DFHRESPONSE contiene un sobre SOAP completo y todo su contenido en la página de códigos UTF-8.

El contenedor DFHRESPONSE está presente cuando se llama a un manejador de mensajes, y el contenedor DFHFUNCTON contiene SEND-RESPONSE o RECEIVE-RESPONSE.

En esta situación, el protocolo normal es devolver DFHRESPONSE a la interconexión con el mismo contenido o con contenido modificado. El proceso de interconexión continúa normalmente, con el siguiente programa de manejador de mensajes en la interconexión, si hubiera uno.

El contenedor DFHRESPONSE también está presente, con una longitud de cero, cuando DFHFUNCTON contiene RECEIVE-REQUEST, SEND-REQUEST, PROCESS-REQUEST o HANDLER-ERROR.

Contenedor DFHWS-CCSID:

DFHWS-CCSID es un contenedor de DATATYPE(BIT) que contiene una palabra completa (4 bytes) que especifica el CCSID de los datos en el contenedor de respuestas.

El contenedor es válido sólo para la interconexión de modalidad de proveedor que utiliza código CICS para transformar la estructura de lenguaje en XML.

El CCSID debe ser compatible con el CCSID utilizado para generar el archivo WSBIND. Si no lo es, la respuesta SOAP que se produce puede contener caracteres incorrectos o no válidos.

El CCSID no se puede cambiar a o desde 930, 1390, 5026 y 1026. Además, CICS no permite que se cambie CCSID a uno que pueda utilizar como cliente CCSID.

Si se producen problemas al procesar el valor en el contenedor DFHWS-CCSID, el proceso continúa utilizando el CCSID desde el archivo WSBIND.

El contenedor DFHWS-CCSID se comprueba únicamente en la devolución de un programa de aplicación basado en el canal.

Cómo controlan los contenedores los protocolos de interconexión

El contenido conjunto de los contenedores DFHFUNCTON, DFHREQUEST y DFHRESPONSE controlan los protocolos de interconexión.

Durante las dos fases de ejecución de una interconexión (la fase de solicitud y la fase de respuesta), el valor de DFHFUNCTION determina qué contenedores de control se pasan a cada manejador de mensajes:

DFHFUNCTION	Contexto	DFHREQUEST	DFHRESPONSE
RECEIVE-REQUEST	Proveedor de servicios; fase de solicitud	Presente (longitud > 0)	Presente (longitud = 0)
SEND-RESPONSE	Proveedor de servicios; fase de respuesta	Ausente	Presente (longitud > 0)
SEND-REQUEST	Solicitante de servicio; fase de solicitud	Presente (longitud > 0)	Presente (longitud = 0)
RECEIVE-RESPONSE	Solicitante de servicio; fase de respuesta	Ausente	Presente (longitud > 0)
PROCESS-REQUEST	Proveedor de servicios; controlador de terminal	Presente (longitud > 0)	Presente (longitud = 0)
HANDLER-ERROR	Proveedor o solicitante de servicio; cualquier fase	Ausente	Presente (longitud = 0)
NO-RESPONSE	Proveedor o solicitante de servicio; fase de respuesta	Ausente	Ausente

El proceso subsiguiente se determina por los contenedores que el manejador de mensajes pasa de vuelta a la interconexión:

Durante la fase de la solicitud

- El manejador de mensajes puede devolver el contenedor DFHREQUEST. El proceso continúa en la fase de solicitud con el siguiente manejador. La longitud de los datos en el contenedor no debe ser cero.
- El manejador de mensajes puede devolver el contenedor DFHRESPONSE. El proceso conmuta a la fase de respuesta y se invoca al mismo manejador con DFHFUNCTION establecido en SEND-RESPONSE en un proveedor de servicios y en RECEIVE-RESPONSE en un solicitante de servicio. La longitud de los datos en el contenedor no debe ser cero.
- El manejador de mensajes puede no devolver ningún contenedor. El proceso conmuta a la fase de respuesta y se invoca al mismo manejador con DFHFUNCTION establecido en NO-RESPONSE.

En el controlador de terminal (sólo proveedor de servicios)

- El manejador de mensajes puede devolver el contenedor DFHRESPONSE. El proceso conmuta a la fase de respuesta y se invoca al manejador anterior con un nuevo valor de DFHFUNCTION (SEND-RESPONSE). La longitud de los datos en el contenedor no debe ser cero.

- El manejador de mensajes puede no devolver ningún contenedor. El proceso conmuta a la fase de respuesta y se invoca al manejador anterior con un nuevo valor de DFHFUNCTON (NO-RESPONSE).

Durante la fase de respuesta

- El manejador de mensajes puede devolver el contenedor DFHRESPONSE. El proceso continúa en la fase de respuesta y se invoca al siguiente manejador. La longitud de los datos en el contenedor no debe ser cero.
- El manejador de mensajes puede no devolver ningún contenedor. El proceso continúa en la fase de respuesta y se invoca al siguiente manejador en la secuencia con un nuevo valor de DFHFUNCTON (NO-RESPONSE).

Importante: Durante la fase de solicitud, el manejador de mensajes puede devolver DFHREQUEST o DFHRESPONSE, pero no ambos. Dado que ambos contenedores están presentes cuando se invoca al manejador de mensajes, debe suprimir uno de ellos.

Esta tabla muestra la acción tomada por la interconexión para todos los valores de DFHFUNCTON y todas las combinaciones de DFHREQUEST y DFHRESPONSE devueltas por cada manejador de mensajes.

DFHFUNCTON	Contexto	DFHREQUEST	DFHRESPONSE	Acción
RECEIVE-REQUEST	Proveedor de servicios; fase de solicitud	Presente (longitud > 0)	Presente	(error)
RECEIVE-REQUEST	Proveedor de servicios; fase de solicitud	Presente (longitud > 0)	Ausente	Invoque al siguiente manejador con la función RECEIVE-REQUEST
RECEIVE-REQUEST	Proveedor de servicios; fase de solicitud	Presente (longitud = 0)	No aplicable	(error)
RECEIVE-REQUEST	Proveedor de servicios; fase de solicitud	Ausente	Presente (longitud > 0)	Conmute a la fase de respuesta e invoque el mismo manejador con la función SEND-RESPONSE
RECEIVE-REQUEST	Proveedor de servicios; fase de solicitud	Ausente	Presente (longitud = 0)	(error)
RECEIVE-REQUEST	Proveedor de servicios; fase de solicitud	Ausente	Ausente	Invoque al mismo manejador con la función NO-RESPONSE
SEND-RESPONSE	Proveedor de servicios; fase de respuesta	No aplicable	Presente (longitud > 0)	Invoque al manejador anterior con la función SEND-RESPONSE
SEND-RESPONSE	Proveedor de servicios; fase de respuesta	No aplicable	Presente (longitud = 0)	(error)
SEND-RESPONSE	Proveedor de servicios; fase de respuesta	No aplicable	Ausente	Invoque al mismo manejador con la función NO-RESPONSE
SEND-REQUEST	Solicitante de servicio; fase de solicitud	Presente (longitud > 0)	Presente (longitud = 0)	(error)
SEND-REQUEST	Solicitante de servicio; fase de solicitud	Presente (longitud > 0)	Ausente	Invoque al siguiente manejador con la función SEND-REQUEST

DFHFUNCTION	Contexto	DFHREQUEST	DFHRESPONSE	Acción
SEND-REQUEST	Solicitante de servicio; fase de solicitud	Presente (longitud = 0)	No aplicable	(error)
SEND-REQUEST	Solicitante de servicio; fase de solicitud	Ausente	Presente (longitud > 0)	Conmute a la fase de respuesta e invoque al manejador anterior con la función RECEIVE-RESPONSE
SEND-REQUEST	Solicitante de servicio; fase de solicitud	Ausente	Presente (longitud = 0)	(error)
SEND-REQUEST	Solicitante de servicio; fase de solicitud	Ausente	Ausente	Invoque al mismo manejador con la función NO-RESPONSE
RECEIVE-RESPONSE	Solicitante de servicio; fase de respuesta	No aplicable	Presente (longitud > 0)	Invoque al manejador anterior con la función RECEIVE-RESPONSE
RECEIVE-RESPONSE	Solicitante de servicio; fase de respuesta	No aplicable	Presente (longitud = 0)	(error)
RECEIVE-RESPONSE	Solicitante de servicio; fase de respuesta	No aplicable	Ausente	Invoque al mismo manejador con la función NO-RESPONSE
PROCESS-REQUEST	Proveedor de servicios; controlador de terminal	No aplicable	Presente (longitud > 0)	Invoque al manejador anterior con la función RECEIVE-RESPONSE
PROCESS-REQUEST	Proveedor de servicios; controlador de terminal	No aplicable	Presente (longitud = 0)	(error)
PROCESS-REQUEST	Proveedor de servicios; controlador de terminal	No aplicable	Ausente	Invoque al mismo manejador con la función NO-RESPONSE
HANDLER-ERROR	Proveedor o solicitante de servicio; cualquier fase	No aplicable	Presente (longitud > 0)	Invoque al manejador anterior con la función SEND-RESPONSE o la función RECEIVE-RESPONSE
HANDLER-ERROR	Proveedor o solicitante de servicio; cualquier fase	No aplicable	Presente (longitud = 0)	(error)
HANDLER-ERROR	Proveedor o solicitante de servicio; cualquier fase	No aplicable	Ausente	Invoque al mismo manejador con la función NO-RESPONSE

Contenedores de contexto

En algunas situaciones, los programas de manejadores de mensajes escritos por el usuario y los programas de proceso de cabeceras necesitan información sobre el contexto en el que se invocan. CICS proporciona esta información en un conjunto de *contenedores de contexto*, que se pasan a los programas.

CICS inicializa el contenido de cada contenedor de contexto, pero, en algunos casos, puede cambiar el contenido en los programas de manejador de mensaje y el programa de proceso de cabeceras. Por ejemplo, en una interconexión de proveedor de servicios en la que el controlador de terminal es uno de los manejadores SOAP proporcionados por CICS, puede cambiar el ID de usuario y el

ID de transacción del programa de aplicación de destino modificando el contenido de los contenedores de contexto adecuados.

Alguna de la información proporcionada en los contenedores se aplica únicamente a un proveedor de servicios, o únicamente a un solicitante de servicio y, por lo tanto, algunos de los contenedores de contexto no están disponibles en ambos.

Contenedor DFH-EXIT-HEADER1:

DFH-EXIT-HEADER1 es un contenedor de DATATYPE(CHAR). Contiene una o más cabeceras SOAP que se añaden a una respuesta desde una aplicación de proveedor de servicios web en CICS.

Los programas que ejecutan una salida de usuario global XWSPRRWO pueden añadir una cabecera a una respuesta SOAP. La cabecera debe ser un SOAP válido y los espacios de nombres deben estar autocontenidos en la cabecera XML. Un programa que pone datos en este contenedor debe comprobar su presencia y añadir la nueva cabecera al final de los datos. Siguiendo esta práctica recomendada, se pueden conducir varios programas al mismo punto de salida si es necesario.

Contenedor DFH-HANDLERPLIST:

DFH-HANDLERPLIST es un contenedor de DATATYPE(CHAR) que se inicializa con el contenido del elemento <handler_parameter_list> adecuado del archivo de configuración de interconexión.

Si no ha especificado la lista de parámetros de manejador en el archivo de configuración de interconexión, el contenedor está vacío; es decir, tiene una longitud de cero.

No puede cambiar el contenido de este contenedor.

Contenedor DFH-SERVICEPLIST:

DFH-SERVICEPLIST es un contenedor de DATATYPE(CHAR) que tiene el contenido del elemento <service_parameter_list> del archivo de configuración de interconexión.

Si no ha especificado la lista de parámetros de servicio en el archivo de configuración de interconexión, el contenedor está vacío; es decir, tiene una longitud de cero.

No puede cambiar el contenido de este contenedor.

Contenedor DFHWS-APPHANDLER:

DFHWS-APPHANDLER es un contenedor de DATATYPE(CHAR) que, en una interconexión de proveedor de servicios, se inicializa con el contenido del elemento <apphandler> del archivo de configuración de interconexión.

En el controlador de terminal de una interconexión que contiene el elemento <apphandler>, los manejadores SOAP proporcionados obtienen el nombre del programa de aplicación de destino de este contenedor.

Puede cambiar el contenido de este contenedor en los manejadores de mensajes o programas de proceso de cabeceras.

CICS no proporciona este contenedor en una interconexión de solicitante de servicio.

Conceptos relacionados:

“Manejadores de aplicación” en la página 92

Un manejador de aplicación es un programa CICS al que se enlaza el controlador de terminal de una interconexión de proveedor de servicio SOAP en el tiempo de ejecución.

Contenedor DFHWS-APPHANCLAS:

DFHWS-APPHANCLAS es un contenedor de DATATYPE(CHAR) que, en una interconexión de proveedor de servicios, se inicializa con el contenido del elemento `<apphandler_class>` del archivo de configuración de interconexión.

En el controlador de terminal de una interconexión basada en Java, los manejadores SOAP proporcionados, `<cics_soap_1.1_handler_java>` y `<cics_soap_1.2_handler_java>`, obtienen el nombre del programa de aplicación de destino desde este contenedor

CICS no proporciona este contenedor en una interconexión de solicitante de servicio.

Conceptos relacionados:

“Manejadores de aplicación” en la página 92

Un manejador de aplicación es un programa CICS al que se enlaza el controlador de terminal de una interconexión de proveedor de servicio SOAP en el tiempo de ejecución.

Referencia relacionada:

“Elemento `<apphandler_class>`” en la página 94

Especifica que el controlador de terminal de la interconexión enlaza con un manejador de aplicación Axis2.

Contenedor DFHWS-DATA:

DFHWS-DATA es un contenedor de DATATYPE(BIT) que se utiliza en aplicaciones de solicitante de servicio y, opcionalmente, en aplicaciones de proveedor de servicios que se despliegan con el asistente de servicios web CICS. Mantiene la estructura de datos de nivel superior que se correlaciona con y desde una solicitud SOAP.

En las aplicaciones de solicitante de servicio, el contenedor DFHWS-DATA debe estar presente cuando el programa solicitante de servicio emite un mandato **EXEC CICS INVOKE SERVICE**. Cuando se emite el mandato, CICS convierte la estructura de datos que está en el contenedor en una solicitud SOAP. Cuando se recibe la respuesta SOAP, CICS la convierte en otra estructura de datos que se devuelve a la aplicación en el mismo contenedor.

En aplicaciones de proveedor de servicios, el contenedor DFHWS-DATA se utiliza de forma predeterminada cuando no especifica el parámetro **CONTID** en los trabajos por lotes DFHLS2WS o DFHWS2LS. CICS convierte el mensaje de solicitud SOAP en la estructura de datos que pasa a la aplicación en el contenedor DFHWS-DATA. La respuesta se guarda en el mismo contenedor y CICS convierte la estructura de datos en un mensaje de respuesta SOAP.

Contenedor DFHWS-FAULT:

DFHWS-FAULT es un contenedor de DATATYPE(BIT) que contiene información sobre el tipo de error SOAP que genera CICS.

El contenedor contiene una palabra clave binaria que indica el tipo de error que se puede utilizar en un proceso adicional para una respuesta de servicio web:

1. El error SOAP más reciente era para un error CICS (por ejemplo, CICS o terminación anómala del usuario).
2. El error SOAP más reciente era para un error de aplicación. El contenedor se suprime cuando emite el mandato EXEC CICS SOAPFAULT DELETE. Si se crea un nuevo o segundo error SOAP, CICS actualiza el nuevo contenedor adecuadamente.

No puede cambiar el contenido de este contenedor.

Contenedor DFHWS-LOCATION:

DFHWS-LOCATION es un contenedor de DATATYPE(CHAR) que contiene la cabecera Ubicación proporcionada cuando la respuesta HTTP era 302, 303 o 307.

Contenedor DFHWS-MEP:

DFHWS-MEP es un contenedor de DATATYPE(BIT) que contiene un valor representativo para el patrón de intercambio de mensajes (MEP) de un mensaje SOAP de entrada o salida. Este valor tiene un byte de longitud.

CICS soporta cuatro patrones de intercambio de mensajes para los solicitantes de servicio y los proveedores de servicios. El patrón de intercambio de mensajes se define en el documento WSDL 2.0 para el servicio web y determina si CICS responde como el proveedor, y si CICS espera una respuesta de un proveedor externo. En la modalidad de solicitante, el tiempo que espera CICS por una respuesta se configura utilizando el recurso PIPELINE.

Si ha utilizado el asistente de servicios web de CICS para desplegar su aplicación, este contenedor se llena mediante CICS:

- En una interconexión de proveedor de servicios, este contenedor se llena mediante el manejador de aplicación DFHPITP cuando recibe el mensaje de entrada desde el controlador de terminal.
- En una interconexión de solicitante de servicio, este contenedor se llena cuando la aplicación utiliza el mandato **INVOKE SERVICE**.

Si la aplicación utiliza el canal DFHPIRT para iniciar la interconexión, la aplicación llena este contenedor. Si el contenedor no está presente o no tiene valor, CICS supone que la solicitud está utilizando el MEP de entrada y salida o de sólo salida, dependiendo de si el contenedor DFHNORESPONSE está presente en el canal.

Este contenedor se llena mediante el programa de manejador de aplicaciones proporcionado, DFHPITP. Si utiliza un manejador de aplicaciones diferente, este contenedor no está disponible para su uso.

Tabla 8. Valores que pueden aparecer en el contenedor DFHWS-MEP

Valor	MEP	URI
1	Sólo entrada	http://www.w3.org/ns/wsdl/in-only

Tabla 8. Valores que pueden aparecer en el contenedor DFHWS-MEP (continuación)

Valor	MEP	URI
2	Entrada-salida	http://www.w3.org/ns/wsdl/in-out
4	Sólo entrada sólida	http://www.w3.org/ns/wsdl/robust-in-only
8	Entrada, opcional salida	http://www.w3.org/ns/wsdl/in-opt-out

Contenedor DFHWS-OPERATION:

DFHWS-OPERATION es un contenedor de DATATYPE(CHAR) que se utiliza normalmente en una aplicación de proveedor de servicios desplegada con el asistente de servicios web CICS. Contiene el nombre de la operación que se especifica en una solicitud SOAP.

En un proveedor de servicios, el contenedor proporciona el nombre de la operación para la que se está llamando a la aplicación. Se llena cuando un manejador de mensajes SOAP proporcionado pasa el control al programa de aplicación de destino, y está visible sólo cuando se llama al programa destino con una interfaz de canal.

En una interconexión de solicitante de servicio, el contenedor mantiene el nombre especificado en la opción OPERATION del mandato **EXEC CICS INVOKE SERVICE**. El contenedor no está disponible para la aplicación que emite el mandato.

Este contenedor se llena mediante el programa de manejador de aplicaciones proporcionado, DFHPITP. Si utiliza un manejador de aplicaciones diferente, este contenedor no está disponible para su uso.

Contenedor DFHWS-PIPELINE:

DFHWS-PIPELINE es un contenedor de DATATYPE(CHAR) que contiene el nombre de la PIPELINE en la que se está ejecutando el programa.

No puede cambiar el contenido de este contenedor.

Contenedor DFHWS-RESPWAIT:

DFHWS-RESPWAIT es un contenedor de DATATYPE(BIT) que contiene un número binario de palabra completa no firmada para representar el tiempo de espera en segundos que se aplica a los mensajes de solicitud y respuesta del servicio web de salida.

El valor de este contenedor se proporciona por medio del atributo RESPWAIT de la definición PIPELINE y lo establece CICS cuando se emite el mandato INVOKE SERVICE. Cualquier valor definido en este contenedor por la aplicación de usuario antes de que se emita el mandato INVOKE SERVICE se ignorará.

Un programa de manejador de mensajes que se invoca durante el proceso de interconexión puede sobrescribir el valor del contenedor DFHWS-RESPWAIT. Si se hace esto, el valor actualizado sólo se utiliza si la definición PIPELINE tiene un atributo RESPWAIT que no se establece en DEFT o queda en blanco. Si la definición PIPELINE tiene el atributo RESPWAIT establecido en DEFT o queda en blanco, el valor de tiempo de espera predeterminado del protocolo de transporte se utiliza siempre, independientemente del valor del contenedor DFHWS-RESPWAIT.

Este contenedor se utiliza únicamente en interconexiones de modalidad de solicitante.

Contenedor DFHWS-SOAPLEVEL:

DFHWS-SOAPLEVEL es un contenedor de DATATYPE(BIT) que contiene información sobre el nivel de SOAP utilizado en el mensaje que está procesando.

El contenedor contiene una palabra completa binaria que indica el nivel de SOAP utilizado para una respuesta o solicitud de servicio web:

- 1 La solicitud o respuesta es un mensaje SOAP 1.1.
- 2 La solicitud o respuesta es un mensaje SOAP 1.2.
- 10 La solicitud o respuesta no es un mensaje SOAP.

No puede cambiar el contenido de este contenedor.

Contenedor DFHWS-TRANID:

DFHWS-TRANID es un contenedor de DATATYPE(CHAR) que se inicializa con el ID de transacción de la tarea en la que se está ejecutando la interconexión.

Si cambia el contenido de este contenedor en una interconexión de proveedor de servicios en la que el controlador de terminal es uno de los manejadores SOAP proporcionados por CICS (y lo hace antes de que se pase el control para el programa de aplicación de destino), la aplicación de destino se ejecuta en una nueva tarea asociada al nuevo ID de transacción.

Las nuevas tareas no se pueden iniciar cuando el controlador de terminal y el manejador de aplicación de una interconexión se ejecutan en el mismo servidor JVM. Por esta razón, si despliega aplicaciones JAX-WS Axis2 en CICS, no se puede utilizar DFHWS-TRANID para cambiar el ID de usuario.

Contenedor DFHWS-URI:

DFHWS-URI es un contenedor de DATATYPE(CHAR) que contiene el URI del servicio.

En una interconexión de proveedor de servicios, CICS extrae el URI relacionado del mensaje de entrada y lo coloca en el contenedor DFHWS-URI.

Por ejemplo, si el URI de los servicios web es `http://example.com/location/address` o `jms://queue?destination=INPUT.QUEUE&targetService=/location/address`, el URI relacionado es `/location/address`.

Si está utilizando Web Services Addressing en la interconexión de solicitante, este contenedor se creará y actualizará en el orden siguiente:

1. Al ejecutarse el mandato `INVOKE SERVICE`, crea el contenedor `DFHWS-URI` y lo inicia con el valor de la dirección de punto final de servicio WSDL. Si se ha utilizado el mandato de `API WSACONTEXT BUILD` para crear un contexto de direccionamiento, no debe especificar los parámetros `URI` o `URIMAP` en el mandato `INVOKE SERVICE`.
2. Cuando se ejecuta el manejador de direccionamiento de servicios web (`DFHWSADH`), si existe un `<wsa:To>` `EPR` en el contexto de direccionamiento

con un URI no anónimo, el URI del contenedor DFHWS-URI queda sobrescrito por el valor del <wsa:To> EPR. El URI anónimo se ignora.

El mensaje SOAP se envía al servicio definido por el URI en DFHWS-URI.

En una interconexión de solicitante de servicio, CICS coloca el URI especificado en el mandato **INVOKE SERVICE** o, si no está, el URI del enlace de servicio web, en el contenedor DFHWS-URI. Puede alterar temporalmente este URI mediante un manejador de mensajes en la interconexión.

Un servicio puede utilizar un URI de HTTP, HTTPS, JMS o WebSphere MQ para servicios externos. Un servicio también puede utilizar un URI CICS para un servicio proporcionado por otra aplicación CICS:

URI	Serie de consulta	Descripción
cics://PROGRAM/ <i>programa</i>	? <i>opciones</i>	El manejador de transporte de CICS utiliza un mandato EXEC CICS LINK PROGRAM para enlazar al programa especificado, pasando el canal y contenedores actuales. No se realiza transformación de datos alguna en los datos de la aplicación.
cics://SERVICE/ <i>servicio</i>	?targetServiceUri= <i>targetServiceUri</i> & <i>opciones</i>	El manejador de transporte de CICS utiliza la vía de acceso del servicio, expresada como <i>targetServiceUri</i> , para emparejar un recurso URIMAP para que ejecute la solicitud a través de la interconexión del proveedor. Debe especificar un valor para el parámetro targetServiceUri si utiliza este tipo de URI.
cics://PIPELINE/ <i>interconexión</i>	?targetServiceUri= <i>targetServiceUri</i>	El manejador de transporte de CICS inicia otra interconexión de solicitante de servicio.

Puede añadir parámetros a cada tipo de URI de CICS utilizando el formato *parámetro=valor*, donde cada parámetro está separado por el carácter &. Las reglas siguientes corresponden al URI de CICS:

- El primer parámetro de la serie de consulta debe llevar como prefijo un signo de interrogación. No puede utilizar un signo de interrogación antes de este punto del URI.
- Para incluir un carácter & en un valor de parámetro, debe escapar del carácter. Para obtener más información, consulte la sección de ejemplo al final de este tema.
- CICS cambia los valores en minúsculas de *program* y *pipeline* por mayúsculas.

Los parámetros de la serie de consulta determinan cómo CICS procesa la solicitud al final de la interconexión del solicitante:

maxCommareaLength=valor

Especifique el tamaño máximo de COMMAREA en bytes, que es necesario para el programa de aplicación de destino. El valor no debe sobrepasar 32 763. Si este parámetro está presente en la serie de consulta, CICS enlaza al

programa especificado utilizando COMMAREA. Si este parámetro no está presente en la serie de consulta, CICS enlaza con el programa especificado utilizando un canal.

Este parámetro no es sensible a mayúsculas y minúsculas y solo es válido para el URI `cics://PROGRAM`.

newTask=yes|no

Especifique si el manejador de transporte ejecutará la petición como una tarea nueva.

Este parámetro no distingue entre mayúsculas y minúsculas.

`cics://PROGRAM/testapp?newTask=yes` y `cics://PROGRAM/testapp?NEWTASK=Yes` son lo mismo.

targetServiceUri=uri

Especifique la vía de acceso del servicio al que debe llamarse. En un tipo de destino SERVICE, el manejador de transporte utiliza el valor con `host=localhost` para localizar el recurso URIMAP para iniciar una interconexión de proveedor de servicios. En un tipo de destino PIPELINE, el manejador de transporte utiliza el valor para iniciar otra interconexión de solicitante.

Este parámetro distingue entre mayúsculas y minúsculas.

transid=char(4)

Especifique una transacción bajo la que se ejecutará la petición. El manejador de transporte inicia una secuencia de petición utilizando el ID de transacción especificado.

Este parámetro distingue entre mayúsculas y minúsculas.

userid=char(8)

Especifique un ID de usuario bajo el que se ejecutará la petición. El manejador de transporte inicia una secuencia de petición utilizando el ID de usuario especificado.

Este parámetro no distingue entre mayúsculas y minúsculas.

Tipo de destino	Parámetros en el URI	
PROGRAM	IDusuario	Opcional
PROGRAM	transid	Opcional
PROGRAM	maxCommareaLength	Opcional
PROGRAM	newTask	Opcional, Debe ser sí, o no especificarse si especifica userid o transid .
PROGRAM	targetServiceUri	No soportada
SERVICE	userid	Opcional
SERVICE	transid	Opcional
SERVICE	maxCommareaLength	No soportada
SERVICE	newTask	Opcional, Debe ser sí, o no especificarse si especifica userid o transid .
SERVICE	targetServiceUri	Requerida
PIPELINE	userid	No soportada
PIPELINE	transid	No soportada

Tipo de destino	Parámetros en el URI	
PIPELINE	maxCommareaLength	No soportada
PIPELINE	newTask	No soportada
PIPELINE	targetServiceUri	Requerida

Ejemplos de URI de CICS

En este primer ejemplo, el contenedor DFHWS-URI tiene el siguiente URI al llegar al final de la interconexión:

```
cics://PROGRAM/testapp?newTask=yes&userid=user1
```

El manejador de transporte enlaza al programa CICS denominado testapp, pasando en canal y los contenedores. No se realiza transformación de datos alguna, por lo que el programa destino debe poder procesar el contenido de los contenedores en el canal actual. CICS enlaza con el programa bajo una unidad de trabajo nueva y un ID de usuario distinto a user1.

En este segundo ejemplo, el contenedor DFHWS-URI tiene el siguiente URI al llegar al final de la interconexión:

```
cics://SERVICE/getStockQuote?targetServiceUri=/stock/getQuote&newTask=yes&userid=user2
```

El manejador de transporte sustituye el URI del contenedor DFHWS-URI por el valor /stock/getQuote, busca el URIMAP utilizando la vía de acceso en el parámetro **targetServiceUri** para resolver el URI, e inicia la interconexión del proveedor bajo una tarea nueva y un ID de usuario distinto.

En este tercer ejemplo, el contenedor DFHWS-URI tiene el siguiente URI al llegar al final de la interconexión:

```
cics://PIPELINE/reqpipeA?targetServiceUri=cics://PROGRAM/testapp?newTask=yes%26userid=user1
```

El manejador de transporte sustituye el URI del contenedor DFHWS-URI por el valor cics://PROGRAM/testapp?newTask=yes&userid=user1 e inicia la interconexión de proveedor denominada reqpipeA, pasando el canal y contenedores actuales. Los caracteres %26 escapan del carácter & por lo que el manejador de transporte coloca todo el URI en el contenedor DFHWS-URI.

Contenedor DFHWS-URI-RESID:

Se trata de un contenedor DATATYPE(CHAR) que se encuentra solo disponible para las aplicaciones conectadas mediante una interconexión JSON.

Este contenedor incluye una copia simplificada de la vía de acceso del URI (RESource IDentifier), en la que el fragmento URI de la vía de acceso que se ha utilizado en la correlación de URIMAP se ha eliminado. Por ejemplo,

Si el URIMAP que coincide con la solicitud entrante tiene como PATH:

```
/JSONServices/CustomDetails/*
```

y el URI de entrada del cliente es:

```
http://www.example.org:10000/JSONServices/CustomDetails/customerNumber/13388?action=query
```

, el contenido de DFHWS-URI-RESID sería:

```
customerNumber/13388
```

Las aplicaciones RESTful JSON podrán utilizar este contenedor para ayudar a identificar el ID de recurso (o clave principal) para los recursos RESTful que se han correlacionado utilizando un URIMAP comodín. Esto debería ser bastante más simple que analizar el contenido de DFHWS-URI.

Nota: Si el atributo PATH del URIMAP coincidente no es de carácter comodín (es decir, contiene la vía de acceso completa al URI), el contenido de este contenedor estará vacío.

Nota: El atributo PATH del URIMAP coincidente puede contener un fragmento de serie de consulta opcional. En este caso, el fragmento de serie de consulta se omite al construir este contenedor.

Contenedor DFHWS-URI-QUERY:

Es un contenedor de DATATYPE(CHAR), disponible para los programas de aplicación en todas las interconexiones CICS de modalidad de proveedor HTTP.

Este contenedor incluye el fragmento de serie de consulta del URI. Por ejemplo,

Si el URI de entrada del cliente ha sido:

`http://www.example.org:10000/JSONServices/CustomDetails/customerNumber/13388?action=query&page=1`

el contenido de DFHWS-URI-QUERY sería:

`action=query&page=1`

Las aplicaciones pueden analizar el contenido de este contenedor para identificar parámetros **name=value** individuales del URI.

Nota: Si el URI de entrada no incluye una serie de consulta, este contenedor no estará presente en el canal.

Contenedor DFHWS-URIMAPPATH:

Se trata de un contenedor DATATYPE(CHAR) e incluye una copia de los datos PATH procedentes del URIMAP que se ha utilizado para coincidir con el URI de entrada.

Cualquier interconexión conectada a la aplicación puede utilizar este contenedor para obtener más información sobre cómo se ha conectado.

Contenedor DFHWS-USERID:

DFHWS-USERID es un contenedor de DATATYPE(CHAR) que se inicializa con el ID de usuario de la tarea en la que se está ejecutando la interconexión.

Si cambia el contenido de este contenedor en una interconexión de proveedor de servicios en la que el controlador de terminal es uno de los manejadores SOAP proporcionados por CICS (y lo hace antes de que se pase el control para el programa de aplicación de destino), la aplicación de destino se ejecuta en una nueva tarea asociada al nuevo ID de usuario. A menos que cambie el contenido del contenedor DFHWS-TRANID, la nueva tarea tiene el mismo ID de transacción que la tarea en la que se está ejecutando la interconexión.

Las nuevas tareas no se pueden iniciar cuando el controlador de terminal y el manejador de aplicación de una interconexión se ejecutan en el mismo servidor

JVM. Por esta razón, si despliega aplicaciones JAX-WS Axis2 en CICS, no se puede utilizar DFHWS-USERID para cambiar el ID de usuario.

Contenedor DFHWS-WEBSERVICE:

DFHWS-WEBSERVICE es un contenedor de DATATYPE(CHAR) que se utiliza únicamente en una interconexión de proveedor de servicios. Contiene el nombre del servicio web que especifica el entorno de ejecución cuando la aplicación de destino se ha desplegado utilizando el asistente de servicios web.

CICS no proporciona este contenedor en una interconexión de solicitante de servicio.

Contenedor DFHWS-CID-DOMAIN:

DFHWS-CID-DOMAIN es un contenedor de DATATYPE(CHAR). Contiene el nombre de dominio que se utiliza para generar los valores contenido-ID para hacer referencia a archivos adjuntos binarios.

El valor del nombre de dominio es `cicsts` de forma predeterminada. Puede sobrescribir el valor especificando el elemento `<mime_options>` en el archivo de configuración de interconexión.

No puede cambiar el contenido de este contenedor.

Contenedor DFHWS-MTOM-IN:

DFHWS-MTOM-IN es un contenedor de DATATYPE(BIT) que contiene información sobre las opciones especificadas para el elemento `<cics_mtom_handler>` del archivo de configuración de interconexión e información sobre el formato de mensaje que se ha recibido en la interconexión.

Contiene información para procesar un mensaje MTOM de entrada en la interconexión. El mensaje de entrada puede ser un mensaje de solicitud de un solicitante de servicio web o un mensaje de respuesta de un proveedor de servicios web.

Si no especifica un elemento `<cics_mtom_handler>` en el archivo de configuración de interconexión, o si se recibe un mensaje SOAP en lugar de un mensaje MTOM, este contenedor no se crea.

Si se configura la seguridad de servicios web en la interconexión, o si se conmuta la validación para un servicio web, CICS puede sustituir el contenido del campo `XOP_MODE` en DFHWS-MTOM-IN cuando se crea el contenedor. Por ejemplo, si configura la interconexión para procesar el contenido de mensajes MTOM en modalidad directa y, después, conmuta la validación para el servicio web, CICS sobrescribe el valor definido en el archivo de configuración de interconexión y establece el proceso XOP para que se ejecute en modalidad de compatibilidad. CICS realiza la sustitución debido a las restricciones en apoyo al proceso de documentos XOP y archivos adjuntos binarios en la interconexión.

No puede cambiar el contenido de este contenedor.

Tabla 9. Estructura del contenedor DFHWS-MTOM-IN

Nombre de campo	Longitud (bytes)	Contenido
MTOM_STATUS	4	Contiene el valor "1", que indica que el mensaje que ha recibido CICS está en formato MTOM.
MTOMNOXOP_STATUS	4	Contiene uno de los valores siguientes: 0 El mensaje MTOM contiene archivos adjuntos binarios. 1 El mensaje MTOM no contiene archivos adjuntos binarios.
XOP_MODE	4	Contiene uno de los valores siguientes: 0 El proceso XOP no tiene lugar. 1 El proceso XOP tiene lugar en modalidad de compatibilidad. 2 El proceso XOP tiene lugar en modalidad directa.

Contenedor DFHWS-MTOM-OUT:

DFHWS-MTOM-OUT es un contenedor de DATATYPE(BIT) que tiene información sobre las opciones especificadas para el elemento <cics_mtom_handler> del archivo de configuración de interconexión.

Contiene la información para procesar un mensaje MTOM de salida en la interconexión, si es un mensaje de respuesta para un solicitante de servicio web o un mensaje de solicitud para un proveedor de servicios web.

Si no especifica un elemento <cics_mtom_handler> en el archivo de configuración de interconexión, o si el elemento <mtom_options> del archivo de configuración de interconexión tiene el atributo send_mtom="no", este contenedor no se crea.

En modalidad de proveedor, este contenedor se crea al mismo tiempo que el contenedor DFHWS-MTOM-IN. Si el elemento <mtom_options> en el archivo de configuración de interconexión tiene el atributo send_mtom="same", el campo MTOM_STATUS se configura para indicar si el solicitante de servicio web desea un mensaje de respuesta SOAP o un MTOM.

Si se configura la seguridad de servicios web en la interconexión, o si se conmuta la validación para un servicio web, CICS puede cambiar el campo XOP_MODE de DFHWS-MTOM-OUT cuando se crea el contenedor. Por ejemplo, si configura la interconexión para procesar el documento XOP y archivos adjuntos binarios utilizando la modalidad directa y, a continuación, conmuta la validación para un servicio web, CICS sustituye el valor definido en el archivo de configuración de interconexión y establece el proceso XOP para que se ejecute en modalidad de compatibilidad cuando crea el contenedor. CICS realiza la sustitución debido a las restricciones en apoyo al proceso de documentos XOP y archivos adjuntos binarios en la interconexión.

No puede cambiar el contenido de este contenedor.

Tabla 10. Estructura del contenedor DFHWS-MTOM-OUT

Nombre de campo	Longitud (bytes)	Contenido
MTOM_STATUS	4	Indica si MTOM está habilitado: 0 MTOM no está habilitado. El mensaje de salida se envía en formato SOAP. 1 MTOM está habilitado. El mensaje de salida se envía en formato MTOM.
MTOMNOXOP_STATUS	4	Indica si utilizar MTOM cuando no hay archivos adjuntos binarios: 0 No se envía un mensaje MTOM cuando no hay archivos adjuntos binarios. 1 Se envía un mensaje MTOM cuando no hay archivos adjuntos binarios.
XOP_MODE	4	Indica qué proceso XOP debe tener lugar: 0 Ningún proceso XOP tiene lugar. 1 El proceso XOP tiene lugar en modalidad de compatibilidad. 2 El proceso XOP tiene lugar en modalidad directa.

Contenedor DFHWS-WSDL-CTX:

DFHWS-WSDL-CTX es un contenedor de DATATYPE(CHAR) que se utiliza en una aplicación de solicitante de servicio o de proveedor de servicios desplegada con el asistente de servicios web CICS. Contiene la información de contexto WSDL que se puede utilizar para supervisar propósitos.

DFHWS-WSDL-CTX contiene la siguiente información de contexto para el documento WSDL:

- El nombre y el espacio de nombres de la operación para la que se está invocando la aplicación.
- Si se conocen, el nombre y el espacio de nombres para el puerto WSDL 1.1 o punto final WSDL 2.0 que se está utilizando.

Estos valores se separan por caracteres de espacio. DFHWS-WSDL-CTX se llena mediante CICS sólo a nivel de tiempo de ejecución 2.1 y posterior.

Si ha utilizado el asistente de servicios web de CICS para desplegar su aplicación, este contenedor se llena mediante CICS:

- En una interconexión de proveedor de servicios, este contenedor se llena mediante el manejador de aplicación DFHPITP cuando recibe el mensaje de entrada desde el controlador de terminal.
- En una interconexión de solicitante de servicio, este contenedor se llena cuando la aplicación utiliza el mandato **INVOKE SERVICE**.

Si la aplicación utiliza el programa DFHPIRT para iniciar la interconexión, la aplicación llena el contenedor DFHWS-WSDL-CTX si es necesario.

Contenedor DFHWS-XOP-IN:

DFHWS-XOP-IN es un contenedor de DATATYPE(BIT). Contiene una lista de referencias a los archivos adjuntos binarios que se han desempaquetado de un mensaje MIME de entrada y se han colocado en contenedores utilizando el proceso XOP.

Cada registro de archivo adjunto en el contenedor DFHWS-XOP-IN consta de estos elementos:

- El nombre de 16 bytes del contenedor que contiene las cabeceras MIME para el archivo adjunto binario
- El nombre de 16 bytes del contenedor que contiene el archivo adjunto binario
- La longitud de 2 bytes del ID de contenido, en formato binario de media palabra firmado
- El ID de contenido, incluidos los delimitadores < y >, almacenados como una serie de caracteres ASCII

No puede cambiar el contenido de este contenedor.

Contenedor DFHWS-XOP-OUT:

DFHWS-XOP-OUT es un contenedor de DATATYPE(BIT). Contiene una lista de referencias a los contenedores que contienen archivos adjuntos binarios. Los archivos adjuntos binarios están empaquetados en un mensaje MIME de salida por el programa del manejador MTOM.

Cada registro de archivo adjunto en el contenedor DFHWS-XOP-OUT consta de estos elementos:

- El nombre de 16 bytes del contenedor que contiene las cabeceras MIME para el archivo adjunto binario
- El nombre de 16 bytes del contenedor que contiene el archivo adjunto binario
- La longitud de 2 bytes del ID de contenido, en formato binario de media palabra firmado
- El ID de contenido, incluidos los delimitadores < y >, almacenados como una serie de caracteres ASCII

No puede cambiar el contenido de este contenedor.

Contenedores del programa de proceso de cabeceras

Los manejadores de mensajes SOAP 1.1 y SOAP 1.2 proporcionados por CICS enlazan con los programas de proceso de cabeceras utilizando el canal DFHHHC-V1. Los contenedores que se pasan al canal incluyen varios contenedores que son específicos para la interfaz del programa de proceso de cabeceras, y conjuntos de *contenedores de contexto* y *contenedores de usuarios* accesibles para todos los programas de proceso de cabeceras y programas de manejadores de mensajes en la interconexión.

El contenedor DFHHEADER es específico para la interfaz del programa de proceso de cabeceras. Otros contenedores están disponibles en otras partes de la interconexión, pero tienen usos específicos en un programa de proceso de cabeceras. Los contenedores de esta categoría son DFHWS-XMLNS, DFHWS-BODY y DFHXMLSS-PARSE.

Nota: Aunque el servicio web que utiliza Axis2 para procesar mensajes SOAP puede utilizar la interfaz del programa de proceso de cabeceras, es más eficaz escribir sus propios manejadores Axis2 en Java para procesar las cabeceras SOAP. Para obtener más información sobre la creación de manejadores Axis2, consulte Escribir su propio módulo Axis2

Contenedor DFHHEADER

Cuando se invoca el programa de proceso de cabeceras, DFHHEADER contiene el único bloque de cabecera que ha provocado que el programa de proceso de cabeceras se realice. Cuando se especifica el programa de cabecera con `<mandatory>true</mandatory>` o `<mandatory>1</mandatory>` en el archivo de configuración de interconexión, se invoca incluso cuando no hay bloque de cabecera coincidente en el mensaje SOAP. En este caso, el contenedor DFHHEADER tiene una longitud de cero. Este es el caso cuando se invoca un proceso de cabeceras para añadir un bloque de cabecera a un mensaje SOAP que no tiene bloques de cabecera.

El mensaje SOAP que crea CICS no tiene cabeceras inicialmente. Si desea añadir cabeceras al mensaje, debe asegurarse de que se invoca al menos un programa de proceso de cabeceras, especificando `<mandatory>true</mandatory>` o `<mandatory>1</mandatory>`.

Cuando el programa de cabeceras regresa, el contenedor DFHHEADER debe contener cero o más bloques de cabecera que CICS inserta en el mensaje SOAP en lugar del original:

- Puede devolver el bloque de cabecera original sin modificar.
- Puede modificar el contenido del bloque de cabecera.
- Puede añadir uno o más bloques de cabecera nuevos al bloque original.
- Puede sustituir el bloque de cabecera original por uno o varios bloques diferentes.
- Puede suprimir el bloque de cabecera completamente.

Contenedor DFHWS-XMLNS

Cuando se invoca el programa de proceso de cabeceras, DFHWS-XMLNS contiene información sobre los espacios de nombres XML que se declaran en el sobre SOAP. El programa de cabeceras puede utilizar esta información para realizar las siguientes tareas:

- Resolver nombres calificados que encuentra en el bloque de cabecera
- Construir nombres calificados en bloques de cabecera nuevos o modificados.

La información del espacio de nombres consta de una lista de declaraciones de espacio de nombres, que utiliza la notación XML estándar para declarar espacios de nombres. Las declaraciones de espacio de nombres en DFHWS-XMLNS están separadas por espacios. Por ejemplo:

```
xmlns:na='http://abc.example.org/schema' xmlns:nx='http://xyz.example.org/schema'
```

Puede añadir más declaraciones de espacio de nombres al sobre SOAP añadiéndolas al contenido de DFHWS-XMLNS. Sin embargo, los espacios de nombres cuyo alcance es un bloque de cabecera SOAP o un cuerpo SOAP se declaran mejor en el bloque de cabecera o el cuerpo respectivamente. Se recomienda no suprimir declaraciones de espacio de nombres desde el contenedor DFHWS-XMLNS en un programa de proceso de cabeceras porque los elementos

XML que no están visibles en el programa pueden basarse en ellas.

Contenedor DFHWS-BODY

Este contenedor contiene la sección de cuerpo del sobre SOAP. El programa de proceso de cabeceras puede modificar el contenido.

Cuando se invoca el programa de proceso de cabeceras, DFHWS-BODY contiene el elemento SOAP <Body>.

Cuando regresa el programa de cabeceras, el contenedor DFHWS-BODY debe contener de nuevo un <Body> SOAP válido, que CICS inserta en el mensaje SOAP en lugar del original:

- Puede devolver el cuerpo original sin modificar.
- Puede modificar el contenido del cuerpo.

No debe suprimir el cuerpo SOAP por completo, ya que todos los mensajes SOAP deben contener un elemento <Body>.

Contenedor DFHXMLSS-PARSE

Cuando utiliza los elementos <cics_soap_1.1_handler> o <cics_soap_1.2_handler> en la configuración de interconexión, y se invoca el programa de cabeceras, DFHXMLSS-PARSE contiene los registros de XML System Services (XMLSS) para esa cabecera. Este contenedor no se crea cuando se utilizan los elementos <cics_soap_1.1_handler_java> o <cics_soap_1.2_handler_java>.

Control, contexto y contenedores de usuario

Siempre que se describen los contenedores, la interfaz pasa los *contenedores de control*, *contenedores de contexto*, y *contenedores de usuarios* al canal DFHHHC-V1.

Para obtener más información acerca de estos contenedores, consulte “Contenedores utilizados en la interconexión” en la página 150.

Contenedores de seguridad

Los contenedores de seguridad se utilizan en el canal DFHWSTC-V1 para enviar y recibir señales de identidad de un Security Token Service (STS) como Tivoli Federated Identity Manager. Esta interfaz se denomina *Interfaz de cliente de confianza* y se puede utilizar en interconexiones de proveedor y solicitante de servicio web.

Contenedor DFHWS-IDTOKEN:

DFHWS-IDTOKEN es un contenedor de DATATYPE(CHAR). Contiene la señal que valida o utiliza el Security Token Service (STS) para emitir una señal de identidad para el mensaje.

La señal debe estar en formato XML.

Utilice este contenedor sólo con el canal DFHWSTC-V1 para la interfaz de cliente de confianza.

Contenedor DFHWS-RESTOKEN:

DFHWS-RESTOKEN es un contenedor de DATATYPE(CHAR). Contiene la respuesta del Security Token Service (STS).

La respuesta depende de la acción que se ha solicitado desde el STS en el contenedor DFHWS-STSACTION.

- Si se emite la acción, este contenedor mantiene la señal que ha intercambiado el STS para la que se ha enviado en el contenedor DFHWS-IDTOKEN.
- Si la acción es válida, este contenedor mantiene un URI para indicar si la señal de seguridad que se ha enviado en el contenedor DFHWS-IDTOKEN es válida o no. Los URI que se pueden devolver son los siguientes:

URI	Descripción
http://schemas.xmlsoap.org/ws/2005/02/trust/status/valid	La señal de seguridad es válida.
http://schemas.xmlsoap.org/ws/2005/02/trust/status/invalid	La señal de seguridad no es válida.

Este contenedor se devuelve en el canal DFHWSTC-V1 cuando se utiliza la interfaz de cliente de confianza.

Contenedor DFHWS-SERVICEURI:

DFHWS-SERVICEURI es un contenedor de DATATYPE(CHAR). Contiene el URI que utiliza Security Token Service (STS) como ámbito AppliesTo.

El ámbito AppliesTo se utiliza para determinar el servicio web con el que está asociada la señal de seguridad.

Utilice este contenedor sólo con el canal DFHWSTC-V1 para la interfaz de cliente de confianza.

Contenedor DFHWS-STSACTION:

DFHWS-STSACTION es un contenedor de DATATYPE(CHAR). Contiene el URI de la acción que toma el Security Token Service (STS) para validar o emitir una señal de seguridad.

Los valores de URI que puede especificar en este contenedor son los siguientes:

URI	Descripción
http://schemas.xmlsoap.org/ws/2005/02/trust/Issue	STS emite una señal a cambio de la que se ha enviado en el contenedor DFHWS-IDTOKEN.
http://schemas.xmlsoap.org/ws/2005/02/trust/Validate	STS valida la señal que se ha enviado en el contenedor DFHWS-IDTOKEN.

Utilice este contenedor sólo con el canal DFHWSTC-V1 para la interfaz de cliente de confianza.

Contenedor DFHWS-STSFault:

DFHWS-STSFault es un contenedor de DATATYPE(CHAR). Contiene el error devuelto por Security Token Service (STS).

Si se produce un error, STS emite un error SOAP. El contenido del error SOAP se devuelve en este contenedor.

Este contenedor se devuelve en el canal DFHWSTC-V1 cuando se utiliza la interfaz de cliente de confianza.

Contenedor DFHWS-STReason:

DFHWS-STReason es un contenedor de DATATYPE(CHAR). Contiene el contenido del elemento <wst:Reason>, si este elemento está presente en el mensaje de respuesta del Security Token Service (STS).

El elemento <wst:Reason> contiene una serie opcional que proporciona información relacionada con el estado de la solicitud de validación que se ha enviado CICS a STS. Si la señal de seguridad no es válida, la información proporcionada por el STS en este elemento puede ayudarle a determinar por qué la señal no es válida.

Para obtener más información, consulte la especificación *Web Services Trust Language* publicada en OASIS WS-Trust v1.4 Standard.

Contenedor DFHWS-STURI:

DFHWS-STURI es un contenedor de DATATYPE(CHAR). Contiene el URI absoluto del Security Token Service (STS) que se utiliza para validar o emitir una señal de identidad para el mensaje SOAP.

El formato del URI es `http://www.example.com:8080/TrustServer/SecurityTokenService`. Puede utilizar HTTP o HTTPS, dependiendo de los requisitos de seguridad.

Utilice este contenedor sólo con el canal DFHWSTC-V1 para la interfaz de cliente de confianza.

Contenedor DFHWS-TOKENType:

DFHWS-TOKENType es un contenedor de DATATYPE(CHAR). Contiene el URI del tipo de señal solicitado que emite el Security Token Service (STS) como una señal de identidad para el mensaje SOAP.

Puede especificar cualquier tipo de señal válido, pero debe ser compatible con STS.

Utilice este contenedor sólo con el canal DFHWSTC-V1 para la interfaz de cliente de confianza.

Contenedores de soporte SAML

Los contenedores de sólo lectura que utiliza el soporte CICS SAML.

En los temas siguientes, *nnn* significa que puede haber más de un contenedor. Los contenedores se numeran de 001 a *nnn* (el número de contenedores de este tipo devueltos). No se soportan más de 999 contenedores de un tipo concreto y los

datos en la aserción SAML que están relacionados con ellos se ignoran. Los contenedores que no se correlacionan con un DSECT son de longitud variable.

Contenedor DFHSAML-AnnnVmmm:

DFHSAML-AnnnVmmm es un contenedor de DATATYPE(CHAR). Contiene el Valor de atributo *mmm* para el atributo *nnn*, donde *nnn* y *mmm* son números de tres dígitos.

El número de atributos es SAMLC-ATTRNUM en el contenedor DFHSAML-COUNTS.

El número de valores para este atributo está en DFHSAML-ATTRAnnn.

Contenedor DFHSAML-ASSQNAME:

DFHSAML-ASSQNAME es un contenedor de DATATYPE(CHAR). Contiene el espacio de nombres Aserción SAML.

Los valores posibles son

SAML 1.1

urn:oasis:names:tc:SAML:1.0:assertion

SAML 2.0

urn:oasis:names:tc:SAML:2.0:assertion

Esta aserción debe ser un URI. Si la aserción es más compleja, se extrae en 3 partes.

Contenedor DFHSAML-ATTRAnnn:

DFHSAML-ATTRAnnn es un contenedor de DATATYPE(BIT). Contiene un campo BIN(31) con el número de valores para el atributo *nnn*. El número máximo de valores es 999.

El número de atributos es SAMLC-ATTRNUM en el contenedor DFHSAML-COUNTS.

Contenedor DFHSAML-ATTRFnnn:

DFHSAML-ATTRFnnn es un contenedor de DATATYPE(CHAR). Contiene el formato de nombre de atributo para el atributo *nnn*, donde *nnn* es un número de 3 dígitos.

El número de atributos es SAMLC-ATTRNUM en el contenedor DFHSAML-COUNTS.

Contenedor DFHSAML-ATTRNnnn:

DFHSAML-ATTRNnnn es un contenedor de DATATYPE(CHAR). Contiene el nombre de atributo para el atributo *nnn*, donde *nnn* es un número de 3 dígitos.

El número de atributos es SAMLC-ATTRNUM en el contenedor DFHSAML-COUNTS.

Contenedor DFHSAML-ATTRSnnn:

DFHSAML-ATTRSnnn es un contenedor de DATATYPE(CHAR). Contiene el espacio de nombre de atributo para el atributo *nnn*, donde *nnn* es un número de 3 dígitos.

El número de atributos es SAMLC-ATTRNUM en el contenedor DFHSAML-COUNTS.

Contenedor DFHSAML-ATTRYnnn:

DFHSAML-ATTRYnnn es un contenedor de DATATYPE(CHAR). Contiene el nombre descriptivo de atributo para el atributo *nnn*, donde *nnn* es un número de 3 dígitos.

El número de atributos es SAMLC-ATTRNUM en el contenedor DFHSAML-COUNTS.

Contenedor DFHSAML-AUDNRnnn:

DFHSAML-AUDNRnnn es un contenedor de DATATYPE(CHAR). Contiene el nombre AudienceRestriction.

El número de contenedores devueltos es AUDNRNUM.

Contenedor DFHSAML-AUTHMETH:

DFHSAML-AUTHMETH es un contenedor de DATATYPE(CHAR). Contiene el método que se utiliza para autenticar el poseedor de la señal.

Los métodos incluyen password, Kerberos y ltpa.

Contenedor DFHSAML-CERTIDN:

DFHSAML-CERTIDN es un contenedor de DATATYPE(CHAR). Contiene el nombre distinguido del emisor del certificado X.509 del firmante de SAML.

Contenedor DFHSAML-CERTSDN:

DFHSAML-CERTSDN es un contenedor de DATATYPE(CHAR). Contiene el nombre distinguido del asunto del certificado X.509 del firmante de SAML.

Contenedor DFHSAML-CERTSNUM:

DFHSAML-CERTSNUM es un contenedor de DATATYPE(CHAR). Es un campo de ocho caracteres que contiene el número de serie del certificado X.509 del firmante SAML.

Contenedor DFHSAML-CONFMETH:

DFHSAML-CONFMETH es un contenedor de DATATYPE(CHAR). Contiene el método SubjectConfirmation que se utiliza en esta señal SAML.

Los métodos válidos son titular de la llave, portador, emisor-garantía. La serie devuelta se basa en el perfil de la señal OASIS SAML 1.1 y 2.0.

Nota: Las señales SAML que tienen más de un método de confirmación no se soportan. Si hay más de un método de confirmación, los resultados son imprevisibles.

Contenedor DFHSAML-COUNTS:

DFHSAML-COUNTS es un contenedor de DATATYPE(BIT). Contiene el número de salidas de contenedores de longitud variable.

Contenedor DFHSAML-FLAGS:

DFHSAML-FLAGS es un contenedor de DATATYPE(CHAR). Contiene una recopilación de bytes de dispositivo.

Contenedor DFHSAML-ISSUER:

DFHSAML-ISSUER es un contenedor de DATATYPE(CHAR). Contiene el nombre del emisor.

Contenedor DFHSAML-NAMID:

DFHSAML-NAMID es un contenedor de DATATYPE(CHAR). Contiene el valor de la propiedad de formato de nombre.

Contenedor DFHSAML-NAMIDF:

DFHSAML-NAMIDF es un contenedor de DATATYPE(CHAR). Contiene una referencia a URI que representa la clasificación de información del identificador basada en una serie.

Contenedor DFHSAML-NAMIDQ:

DFHSAML-NAMIDQ es un contenedor de DATATYPE(CHAR). Contiene el dominio administrativo o de seguridad que capacita al nombre

Contenedor DFHSAML-NAMIDSP:

DFHSAML-NAMIDSP es un contenedor de DATATYPE(CHAR). Contiene el identificador de nombre que establece el proveedor de servicios o la afiliación de proveedores de la entidad.

Contenedor DFHSAML-NAMIDSPQ:

DFHSAML-NAMIDSPQ es un contenedor de DATATYPE(CHAR). Contiene el nombre de un proveedor de servicios o afiliación de proveedores.

Contenedor DFHSAML-OUTTOKEN:

DFHSAML-OUTTOKEN es un contenedor de DATATYPE(CHAR). Contiene una señal SAML.

Si es un contenedor de entrada, contiene la señal validada anteriormente, que se está direccionando a otro proveedor de servicios, o ampliando y, después, direccionando.

Si es un contenedor de salida, contiene una salida de la señal SAML por el proceso DFHSAML. Si el proceso es validación o extracción, esta señal es la señal validada, extraída o modificada y renunciada.

Contenedor DFHSAML-PROXYnnn:

DFHSAML-PROXYnnn es un contenedor de DATATYPE(CHAR). Contiene el nombre de audiencia ProxyRestriction.

Contenedor DFHSAML-RESPONSE:

DFHSAML-RESPONSE es un contenedor de DATATYPE(BIT). Contiene un código de respuesta que se utiliza internamente.

Contenedor DFHSAML-SAMLID:

DFHSAML-SAMLID es un contenedor de DATATYPE(CHAR). Contiene una serie que representa el ID para SAML 2.0, o AssertionID para SAML 1.1.

Contenedor DFHSAML-SUBJADDR:

DFHSAML-SUBJADDR es un contenedor de DATATYPE(CHAR). Contiene la dirección IP en SubjectLocality.

Restricción: Este contenedor no se devuelve para SAML 2.0.

Contenedor DFHSAML-SUBJDNS:

DFHSAML-SUBJDNS es un contenedor de DATATYPE(CHAR). Contiene la DNSAddress en SubjectLocality.

Contenedor DFHSAML-TIMES:

DFHSAML-TIMES es un contenedor de DATATYPE(CHAR). Contiene una recopilación de valores de tiempo.

Contenedores generados por CICS

CICS genera contenedores para almacenar datos como matrices de variables y series largas. Estos contenedores se crean durante el proceso de interconexión y se utilizan como entrada a, o salida del programa de aplicación. Estos contenedores tienen el prefijo DFH.

El convenio de denominación para estos contenedores es utilizar el módulo CICS que los ha creado, combinado con un sufijo numérico que hace que el nombre del contenedor sea exclusivo en la solicitud. Estos nombres de contenedor tienen lugar durante el proceso de interconexión:

DFHPIAXIS-*nnnnnnnn*

Contenedores que se utilizan para almacenar series y matrices de variables que se pasan a la aplicación en interconexiones Axis2. Este contenedor también puede incluir datos binarios.

DFHPICC-*nnnnnnnnnn*

Contenedores que se utilizan para almacenar series y matrices de variables que se pasan a la aplicación. Este contenedor también puede incluir datos binarios.

DFHPIII-nnnnnnnnn

Contenedores de archivos adjuntos de salida creados cuando la interconexión está habilitada con el manejador de mensajes MTOM y se está ejecutando en modalidad directa. Estos contenedores se crean cuando el programa de aplicación proporciona datos binarios en un campo en lugar de en un contenedor.

DFHPIMM-nnnnnnnnn

Contenedores de archivos adjuntos de entrada creados durante el proceso de mensajes MIME. Estos contenedores los genera CICS cuando el manejador de mensajes MTOM está habilitado en la interconexión. Cuando el proceso de modalidad directa está habilitado, estos contenedores se pueden pasar a través de la aplicación directamente.

DFHPIXO-nnnnnnnnn

Contenedores de archivos adjuntos de salida creados cuando la interconexión está habilitada con el manejador de mensajes MTOM y se está ejecutando en modalidad de compatibilidad.

Los nombres de contenedor numerados comienzan con 1 para cada solicitud de servicio web; por ejemplo, DFHPICC-00000001. Sin embargo, si el programa de aplicación utiliza el mandato **INVOKE SERVICE** para iniciar más de una solicitud de servicio web en el mismo canal, los contenedores que se devuelven a la aplicación para una respuesta pueden existir aún cuando se realizan más solicitudes. En esta situación, CICS comprueba si ya existe el contenedor e incrementa el número del contenedor generado para evitar un conflicto de denominación.

Contenedores de usuarios

Estos contenedores contienen la información que un manejador de mensajes necesita pasar a otro. El uso de los contenedores de usuarios es un asunto de los manejadores de mensajes. Puede elegir sus propios nombres para estos contenedores, pero no debe utilizar nombres que empiezan por DFH.

Proceso de tiempo de ejecución para servicios web

Para enviar una solicitud a un proveedor de servicios web o para recibir una solicitud de un solicitante de servicio web, la aplicación (o programa derivador) debe interactuar correctamente con el soporte de servicios web en CICS. También puede controlar el proceso que tiene lugar en la interconexión para determinar cómo se manejan las solicitudes de entrada y salida.

Cómo invoca CICS un programa proveedor de servicios desplegado con el asistente de servicios web

Cuando se invoca una aplicación de proveedor de servicios que se ha desplegado utilizando el asistente de servicios web CICS, CICS se enlaza con ella con COMMAREA o un canal.

Especifique qué clase de interfaz se utiliza cuando ejecuta el procedimiento JCL DFHWS2LS o DFHLS2WS con el parámetro **PGMINT**. Si especifica un canal, puede poner el nombre del contenedor en el parámetro **CONTID**.

- Si el programa se invoca con una interfaz COMMAREA, COMMAREA contiene la estructura de datos de nivel superior que ha creado CICS desde la solicitud SOAP.
- Si el programa se invoca con una interfaz de canal, la estructura de datos de nivel superior se pasa al programa en el contenedor que se ha especificado en el parámetro **CONTID** de DFHWS2LS o DFHLS2WS. Si no ha especificado el parámetro **CONTID**, los datos se pasan en el contenedor DFHWS-DATA. La

interfaz de canal soporta matrices con varios elementos, que se representan como series de estructuras de datos conectadas en una serie de contenedores. Estos contenedores también estarán presentes.

Cuando codifica los mandatos de API para trabajar con los contenedores, no necesita especificar la opción CHANNEL, porque todos los contenedores están asociados al canal actual (el canal que se ha pasado al programa). Si necesita saber el nombre del canal, utilice el mandato **EXEC CICS ASSIGN CHANNEL**.

Cuando el programa ha procesado la solicitud, debe utilizar el mismo mecanismo para devolver la respuesta: si se ha recibido la solicitud en COMMAREA, entonces, la respuesta debe devolverse en COMMAREA; si la solicitud se ha recibido en un contenedor, la respuesta debe devolverse en el mismo contenedor.

Si se ha encontrado un error cuando el programa de aplicación está emitiendo un mensaje de respuesta, CICS revierte todos los cambios a menos que la aplicación haya llevado a cabo un punto de sincronización.

Si el servicio web proporcionado por el programa no está diseñado para devolver una respuesta, CICS ignorará todo en el COMMAREA o contenedor cuando el programa devuelve algo.

Invocación de un servicio web desde una aplicación desplegada con el asistente de servicios web

Una aplicación de solicitante de servicio que se despliega con el asistente de servicios web utiliza el mandato **EXEC CICS INVOKE SERVICE** para invocar un servicio web. La solicitud y respuesta se correlacionan con una estructura de datos en el contenedor DFHWS-DATA. Este método de invocación de un servicio no se soporta en JSON.

Procedimiento

1. Cree un canal y llénelo con contenedores. Como mínimo, debe estar presente el contenedor DFHWS-DATA. DFHWS-DATA contiene una estructura de datos de alto nivel que CICS convertirá en una solicitud SOAP. Si la solicitud SOAP contiene matrices con varios elementos, se representan como series de estructuras de datos conectadas en una serie de contenedores. Estos contenedores también deben estar presentes en el canal.
2. Invoque el servicio web de destino. Utilice el mandato siguiente:

```
EXEC CICS INVOKE SERVICE(webservice)  
                        CHANNEL(userchannel)  
                        OPERATION(operation)
```

donde:

- *webservice* es el nombre del recurso WEBSERVICE que define el servicio web que se va a invocar. El recurso WEBSERVICE especifica la ubicación de la descripción de servicio web y el archivo de enlace de servicio web que CICS utiliza cuando se comunica con el servicio web.
- *userchannel* es el canal que posee el contenedor DFHWS-DATA y otros contenedores asociados a la estructura de datos de la aplicación.
- *operation* es el nombre de la operación que se va a invocar en el servicio web de destino.

Para obtener más información, consulte el apartado “Optimización local para servicios web” en la página 185.

3. Si el mandato ha funcionado correctamente, recupere los contenedores de respuesta del canal. Como mínimo, debe estar presente el contenedor DFHWS-DATA. Contiene la estructura de datos de nivel superior que ha creado CICS desde la respuesta SOAP. Si la respuesta contiene matrices con varios elementos, se representan como una serie de estructuras de datos conectadas en una serie de contenedores. Estos contenedores estarán presentes en el canal.
4. Si el solicitante de servicio recibe un mensaje de error SOAP desde el servicio web invocado, debe decidir si el programa de aplicación debe revertir algún cambio. Si se produce un error SOAP, se devuelve un error INVREQ con un valor RESP2 de 6 al programa de aplicación. Sin embargo, si la optimización está en vigor, se utiliza la misma transacción en el solicitante y el proveedor. Si se produce un error en un proveedor de servicios web optimizado localmente, todo el trabajo realizado por la transacción se revierte en el proveedor y el solicitante. Se devuelve un error INVREQ al solicitante con un valor RESP2 de 16.

Optimización local para servicios web

Puede utilizar el nombre de aplicación de proveedor en el archivo de enlace de servicio web asociado al recurso WEBSERVICE para habilitar la optimización local de la solicitud de servicio web.

Utilizando el mandato INVOKE SERVICE, puede especificar URIMAP(urimap) o URI(uri) donde el URI es el URI del servicio web que se va a invocar. Si se especifica un URIMAP, CICS utiliza la modalidad de cliente URIMAP indicada para resolver URI. Si estas opciones se omiten, CICS utiliza el URI especificado en la descripción de servicio web (WSDL) desde la que se había generado WEBSERVICE.

Si el WEBSERVICE indicado está desplegado en una modalidad de solicitante PIPELINE, CICS invoca el servicio web remoto. Este es el escenario más típico.

Si el WEBSERVICE indicado está desplegado en una modalidad de proveedor PIPELINE, CICS invoca el servicio localmente. Si utiliza esta optimización, el mandato EXEC CICS INVOKE SERVICE se optimiza para un mandato EXEC CICS LINK. Esto da como resultados beneficios de rendimiento significativos, pero introduce las siguientes limitaciones:

- El PIPELINE no se controla y, por lo tanto, no se utilizan programas de manejador.
- Los contenedores de control PIPELINE no están presentes en el canal. Algunos contenedores están presentes, incluidos los contenedores DFHWS-DATA, DFHWS-OPERATION y DFHWS-URI. Los contenedores que normalmente contienen XML no están presentes; esto incluye contenedores DFH-REQUEST, DFHWS-BODY y DFHWS-XMLNS.
- Las aplicaciones de proveedor y solicitante deben compartir los mismos libros de copia y deben estar implementados en el mismo lenguaje de programación.
- Las aplicaciones de proveedor y solicitante comparte una única unidad de trabajo.
- Si no se espera que el servicio web devuelva una respuesta, el mandato EXEC CICS INVOKE SERVICE no devuelve control a la aplicación hasta después de que el PROGRAM de destino ha finalizado.

Si desea utilizar servicios web optimizados localmente pero requiere que los datos estén procesados a través de una PIPELINE, utilice el formato de cics URI descrito aquí: “Opciones para controlar el proceso de interconexión de solicitante” en la página 190. Este mecanismo es menos eficaz que utilizar el enfoque optimizado

completamente, pero evita el coste de procesamiento de salir a la red.

Limitaciones de tiempo de ejecución para código generado por el asistente de servicios web

En tiempo de ejecución, CICS es capaz de transformar casi cualquier mensaje SOAP válido que se ajusta a la descripción de servicio web (WSDL) en estructuras de datos equivalentes. Sin embargo, hay algunas limitaciones que debe tener en cuenta cuando desarrolla una aplicación de proveedor de servicios o de solicitante de servicio mediante los trabajos por lotes del asistente de servicios web.

Páginas de códigos

CICS puede soportar mensajes SOAP que le envían en cualquier página de códigos si hay un HTTP adecuado o una cabecera WebSphere MQ que identifica la página de códigos. CICS convierte el mensaje SOAP en UTF-8 para procesarlo en la interconexión antes de transformarlo en una página de códigos que necesita el programa de aplicación. Para minimizar el impacto de rendimiento, se recomienda utilizar la página de códigos UTF-8 al enviar mensajes SOAP a CICS. CICS siempre envía mensajes SOAP en UTF-8.

CICS sólo puede transformar mensajes SOAP si los datos de aplicación están codificados en EBCDIC o UTF-16. Las aplicaciones que esperan que los datos se codifiquen en páginas de códigos como UTF-8, ASCII e ISO8859-1 no se soportan. Si desea utilizar caracteres DBCS en sus estructuras de datos y mensajes SOAP, debe especificar una página de códigos que soporte DBCS. La página de códigos EBCDIC que seleccione también debe estar soportada por los servicios de conversión de Java y z/OS. Los servicios de conversión de z/OS también deben configurarse para soportar la conversión de la página de códigos del mensaje SOAP a UTF-8. Consulte *Support for UTF-16 in application data* para obtener más información sobre el soporte UTF-16.

Para establecer una página de códigos adecuada, puede utilizar el parámetro de inicialización del sistema **LOCALCCSID** en el parámetro **CCSID** opcional en los trabajos del asistente de servicios web. Si utiliza el parámetro **CCSID**, el valor que especifica sustituye la página de códigos **LOCALCCSID** para ese servicio web particular. Si no especifica el parámetro **CCSID**, se utiliza la página de códigos **LOCALCCSID** para convertir los datos y el archivo de enlace de servicio web se codifica en US EBCDIC (Cp037).

Contenedores

En la modalidad de proveedor de servicios, si especifica que el parámetro **PGMINT** tiene un valor de **CHANNEL**, el contenedor que tiene los datos de aplicación se debe escribir y leer desde la modalidad binaria. Este contenedor es DFHWS-DATA de forma predeterminada. El mandato **PUT CONTAINER** debe tener la opción **DATATYPE** establecida en **BIT**, o debe omitir la opción **FROMCCSID** para que **BIT** siga siendo el valor predeterminado. Por ejemplo, el código siguiente indica explícitamente que los datos del contenedor **CUSTOMER-RECORD** en el canal actual deben escribirse en modalidad binaria.

```
EXEC CICS PUT CONTAINER (CUSTOMER-RECORD)
              FROM (CREC)
              DATATYPE (BIT)
```

Aunque todos los contenedores están en modalidad **BIT**, los campos de texto dentro de la estructura de lenguaje que correlacionan estos datos deben utilizar una página de códigos EBCDIC - la misma página de códigos que ha especificado

en el parámetro **LOCALCCSID** o **CCSID**. Si está utilizando DFHWS2LS para generar el archivo de enlace de servicio web, puede haber muchos contenedores en el canal que contienen partes de la estructura de datos completa. Si es este el caso, los campos de texto de cada uno de estos contenedores debe leerse y escribirse utilizando la misma página de códigos.

Si el programa de aplicación está llenando contenedores que se van a convertir en mensajes SOAP, la aplicación es responsable de garantizar que los contenedores tienen la cantidad de contenido correcta. Si un contenedor tiene menos datos de los esperados, CICS emite un error de conversión.

Si un programa de aplicación utiliza el mandato **INVOKE SERVICE**, cualquier contenedor que pasa a CICS puede reutilizarse potencialmente y los datos dentro de él sustituirse. Si desea mantener los datos en estos contenedores, cree un nuevo canal y copie en él los contenedores antes de ejecutar el programa. Si tiene un servicio web de modalidad de proveedor que también es un servicio web de modalidad de solicitante, se recomienda que utilice un canal diferente cuando utiliza el mandato **INVOKE SERVICE**, en lugar de utilizar el canal predeterminado al que se había conectado originalmente. Si el programa de aplicación está utilizando el mandato **INVOKE SERVICE** muchas veces, se recomienda que utilice canales diferentes en cada llamada a CICS, o asegurarse de que todos los datos importantes de la primera solicitud se guardan antes de hacer la segunda solicitud.

Ajustarse a la descripción de servicios web

Una descripción de servicio web puede describir parte del posible contenido de un mensaje SOAP como opcional. Si es este el caso, DFHWS2LS asigna campos dentro de la estructura de lenguaje generada para indicar si el contenido está presente o no. En tiempo de ejecución, CICS llena estos campos según corresponda. Si un campo, por ejemplo un indicador de existencia o un campo de aparición, indica que la información no está presente, el programa de aplicación no debe intentar procesar los campos asociados a ese contenido opcional.

Si a un mensaje le falta parte del contenido cuando CICS lo transforma, los campos equivalentes dentro de las estructuras de datos no se inicializan cuando pasan al programa de aplicación.

Una descripción de servicio web también puede especificar las reglas de proceso de espacios en blanco a utilizar cuando se lee un mensaje SOAP, y CICS implementa estas reglas en tiempo de ejecución.

- Si el valor de la faceta `xsd:whiteSpace` es `replace`, los caracteres de espacio en blanco como “pestaña” y “retorno de carro” se sustituyen con espacios.
- Si el valor de la faceta `xsd:whiteSpace` es `collapse`, los caracteres de espacio en blanco finales se eliminan al generar mensajes SOAP. En tiempo de ejecución, los mensajes SOAP de entrada se analizan según la especificación del esquema XML y todos los espacios en blanco de inicio, final e incluidos se eliminan.

Mensajes SOAP

CICS no soporta la derivación de contenido del mensaje SOAP. Por ejemplo, un mensaje SOAP puede utilizar el atributo `xsi:type` para especificar que un elemento tiene un tipo concreto, junto con el atributo `xsi:schemaLocation` para especificar la ubicación del esquema que describe el elemento. CICS no soporta la prestación de la recuperación dinámica del esquema y la transformación del valor del elemento basado en el contenido del esquema. CICS no soporta el atributo `xsi:nil` cuando

el nivel de correlación que se establece en el asistente de servicios web es 1.1 o superior, pero este es el único atributo de instancia de esquema XML que se soporta.

Puede que DFHWS2LS tenga que hacer asunciones sobre el tamaño o longitud máximos de algunos valores en el mensaje SOAP. Por ejemplo, si el esquema no especifica una longitud máxima para un `xsd:string`, DFHWS2LS supone que la longitud máxima es de 255 caracteres y genera una estructura de lenguaje en consecuencia. Puede cambiar este valor utilizando el parámetro **DEFAULT-CHAR-MAXLENGTH** en DFHWS2LS. En tiempo de ejecución, si CICS se encuentra un mensaje SOAP con un valor mayor que el espacio que se ha asignado en la estructura de lenguaje, CICS emite un error de conversión.

Si CICS es el proveedor de servicios, se devuelve un mensaje de error SOAP al solicitante. Si CICS es el solicitante de servicio, se devuelve un código RESP2 desde el mandato **INVOKE SERVICE**.

Algunos caracteres tienen significados especiales en XML, como los caracteres `<` y `>`. Si cualquiera de estos caracteres especiales aparece en una matriz de caracteres procesada por CICS en tiempo de ejecución, se sustituye por la entidad equivalente. Las entidades XML que se soportan son:

Carácter	Entidad XML
&	&
<	<
>	>
"	"
'	'

CICS también soporta los formatos canónicos de las referencias de caracteres numéricos utilizados para los códigos de espacios en blanco:

Carácter	Entidad XML
Pestaña		
Retorno de carro	

Salto de línea	

Tenga en cuenta que este soporte no se extiende a ningún programa de manejador de interconexión que se invoca.

El carácter nulo (`x'00'`) no es válido en cualquier documento XML. Si un campo de tipo de carácter proporcionado por el programa de aplicación contiene el carácter nulo, CICS trunca los datos en ese momento. Esto le permite tratar las matrices de caracteres como series terminadas en nulo. Los campos de tipo de carácter generados por DFHWS2LS desde tipos de datos de esquema XML `base64Binary` o `hexBinary` representan datos binarios y pueden contener caracteres nulos sin truncamiento.

Atención: CICS genera datos XML y JSON desde datos de aplicación estructurados. Si esos datos de aplicación contienen patrones de bits que parecen imágenes XML, JSON, HTML, JPEG preformateadas, o cualquier otro tipo de contenido significativo, CICS no es consciente de este significado semántico y procesa esos datos como texto ordinario o datos binarios. CICS no intenta reconocer patrones en los datos o procesar datos codificados de forma diferente. Por ejemplo, si los datos contienen un XML preformateado (como texto codificado CDATA), esos datos se procesan de la misma forma que otros datos cualquiera. Tenga en cuenta los siguientes datos de aplicación: "An example: <here>". Estos datos proporcionados por la aplicación de ejemplo contienen lo que parece una etiqueta XML, pero se procesará como texto sin formato, dando como resultado la siguiente representación XML: "An example: < here > ". Si la aplicación necesita generar un XML, tenga en cuenta utilizar construcciones xsd:any en sus esquemas XML, o utilizar XML-ONLY=TRUE en los asistentes.

Mensajes de error SOAP

Si CICS es el proveedor de servicios y desea que el programa de aplicación emita un mensaje de error SOAP, utilice el mandato **SOAPFAULT CREATE**. Para utilizar este mandato de API, debe especificar que el parámetro **PGMINT** del asistente de servicios web tiene un valor de CHANNEL. Si no especifica este valor, y el programa de aplicación invoca el mandato **SOAPFAULT CREATE**, CICS no intenta generar un mensaje de respuesta SOAP.

Personalización del proceso de interconexión

Además de proporcionar sus propios manejadores de mensajes, también puede utilizar un conjunto de puntos de salida de usuario globales (GLUEs) para personalizar el proceso que tiene lugar para servicios web de entrada y salida en la interconexión.

Antes de empezar

Debe entender las prácticas recomendadas para escribir programas de salida de usuario global antes de personalizar la interconexión. Si está personalizando una interconexión de proveedor de servicios, debe estar utilizando el manejador de aplicación DFHPITP o Axis2 proporcionado en la interconexión.

Acerca de esta tarea

Puede utilizar las salidas de dominio de interconexión para acceder a los contenedores en una interconexión de proveedor de servicios web, una interconexión de solicitante de servicio web o una interconexión de solicitante de servicio web que contiene un manejador de seguridad. Las salidas de usuario global de interconexión se describen en detalle en Pipeline domain exits.

Procedimiento

1. Seleccione qué puntos de salida de usuario global utilizar:
 - Utilice XWSPRRWI, XWSPRROI, XWSPRROO o XWSPRRWO GLUEs para acceder a contenedores en una interconexión de proveedor de servicios web.
 - Utilice XWSRQRWO, XWSRQROO, XWSROROI o XWSRQRWI GLUEs para acceder a contenedores en una interconexión de solicitante de servicio web.
 - Utilice XWSSRRWO, XWSSRROO, XWSSRROI o XWSSRRWI GLUEs para acceder a contenedores en una interconexión de solicitante de servicio web segura.

2. Utilice el programa de salida de ejemplo DFH\$PIEX para escribir su propio programa de salida de usuario global. DFH\$PIEX está en la biblioteca SDFHSAMP. Se recomienda hacer que el programa sea de hebra segura.
3. Habilite el programa de salida de usuario global.
4. Pruebe el programa de salida de usuario global para asegurarse de que funciona correctamente.

Opciones para controlar el proceso de interconexión de solicitante

En las interconexiones de solicitante de servicio, los manejadores de mensajes pueden determinar dónde se envía la solicitud de servicio web cambiando el URI. CICS proporciona soporte para diferentes formatos URI, por lo tanto tiene mucha más flexibilidad en la forma en la que la interconexión procesa las solicitudes de servicio web.

Cuando la interconexión de solicitante de servicio alcanza el final del proceso, tiene las opciones siguientes:

Enlace a un programa

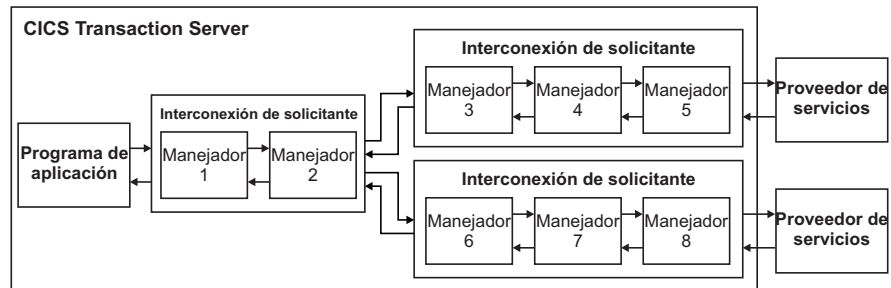
Si cambia el URI al formato `cics://PROGRAM/programa`, donde *programa* es el nombre del programa de aplicación de destino, CICS pasa el canal actual y los contenedores o COMMAREA al programa utilizando un mandato **EXEC CICS LINK**.

Este proceso es similar a la optimización local que se produce cuando las aplicaciones de solicitante de servicio y de proveedor de servicios están en la misma región CICS. Sin embargo, la utilización de este formato de URI proporciona la ventaja de ejecutar primero la solicitud a través de la interconexión y de cualquier manejador de mensajes personalizado. El programa de aplicación de destino debe poder manejar el contenido de los contenedores o COMMAREA.

Inicio de otra interconexión de modalidad de solicitante

Si cambia el URI al formato `cics://PIPELINE/interconexión?targetServiceUri=URI_servicio_destino`, donde *interconexión* es el nombre de un recurso PIPELINE y *URI_servicio_destino* es el URI que desea poner en el contenedor DFHWS-URI, CICS pasa el canal actual y los contenedores a la interconexión de solicitante especificada. Utilice este URI cuando desee enlazar dos más interconexiones de solicitante antes de enviar la solicitud al proveedor de servicios. El número de interconexiones de solicitante que puede encadenar no está limitado.

En el ejemplo siguiente, una interconexión genérica soporta una aplicación. Los manejadores de mensajes 1 o 2 pueden cambiar el URI para cada solicitud dependiendo de los datos de aplicación en el contenedor, enviando la solicitud a una de las dos interconexiones de solicitante que contienen diferentes manejadores de mensajes.



Aunque el ejemplo muestra una única aplicación de solicitante de servicio, muchas aplicaciones pueden utilizar la misma interconexión de solicitante genérica y enviar sus solicitudes a diferentes interconexiones de solicitante antes de que la solicitud se envíe finalmente al proveedor de servicios web adecuado.

Envío directo de la solicitud a la interconexión de modalidad de proveedor

Si cambia el URI al formato `cics://SERVICE/servicio?targetServiceUri=URI_servicio_destino`, donde *servicio* es el nombre del servicio de destino y *URI_servicio_destino* es la vía de acceso al servicio, CICS resuelve la solicitud comparando la vía de acceso a un URIMAP y pasa la solicitud a la interconexión de proveedor correcta. Utilice esta opción cuando desee aprovechar el proceso de la solicitud a través de las interconexiones de solicitante y proveedor sin utilizar la red.

Este URI también puede ser útil cuando las aplicaciones de solicitante y proveedor se escriben en lenguajes diferentes o utilizan diferentes niveles de correlación y esperan datos binarios diferentes.

Control del proceso de interconexión de solicitante utilizando un URI

En las interconexiones de solicitante de servicio, un manejador de mensajes puede determinar dónde se envía la solicitud de servicio web cambiando el URI. Cambiando el formato de URI, puede elegir realizar ciertas optimizaciones, como iniciar otra interconexión de solicitante o iniciar una interconexión de proveedor de servicios sin enviar la solicitud a través de la red.

Antes de empezar

Decida qué opciones desea implementar en la interconexión de solicitante. Consulte “Opciones para controlar el proceso de interconexión de solicitante” en la página 190 para obtener más detalles.

Acerca de esta tarea

La aplicación de solicitante de servicio web puede llenar el contenedor DFHWS-URI utilizando el mandato **EXEC CICS INVOKE SERVICE** o, si la aplicación no proporciona ningún valor, CICS llena el contenedor utilizando el valor en el archivo de enlace de servicio web. Para modificar el URI, debe escribir un manejador de mensajes que cambie el contenido de este contenedor.

Procedimiento

1. Escriba un manejador de mensajes para modificar el contenedor DFHWS-URI según uno de los siguientes formatos de URI:

- Para enlazar con un programa de aplicación, utilice el URI `cics://PROGRAM/program`, donde *program* es el programa de aplicación de destino. No tiene lugar ninguna transformación de datos, por lo que debe asegurarse de que el programa de aplicación puede procesar el contenido de los contenedores en el canal actual. El programa de aplicación puede pasar el canal y los contenedores actuales o un COMMAREA.
- Para iniciar una interconexión de proveedor sin pasar por la red, utilice el URI `cics://SERVICE/service?targetServiceUri=targetServiceUri`, donde *service* es el nombre del servicio y *targetServiceUri* es la vía de acceso del servicio. El manejador de transportes utiliza la vía de acceso del servicio para ubicar el recurso URIMAP que resuelve la solicitud y la pasa a la interconexión de proveedor correcta. CICS no utiliza el nombre del servicio en su proceso.

Se produce un error si no hay instalado un recurso URIMAP para el servicio. La definición de recurso URIMAP también debe especificar `USAGE(PIPELINE)`. El manejador de transporte coloca el valor del parámetro **targetServiceUri** en el contenedor DFHWS-URI e inicia la interconexión de proveedor.

- Para iniciar otra interconexión de solicitante, utilice el URI `cics://PIPELINE/pipeline?targetServiceUri=targetServiceUri`, donde *pipeline* es el nombre del recurso PIPELINE que desea iniciar y *targetServiceUri* es el valor que desea pasar a la siguiente interconexión en el contenedor DFHWS-URI.

Cada tipo de URI tiene parámetros adicionales que puede añadir como una serie de consulta. Para obtener más información sobre el formato de estos URI y las reglas para codificarlos, consulte “Contenedor DFHWS-URI” en la página 166.

2. Utilice un editor XML para añadir el manejador de mensajes al archivo de configuración de interconexión:

```
<service>
  <service_handler_list>
    <handler>
      <program>MYPROG</program>
    </handler>
  </service_handler_list>
</service>
```

3. Inhabilite, descarte o vuelva a instalar el recurso PIPELINE para la interconexión de solicitante para incluir el nuevo programa de manejador de mensajes en la interconexión.
4. Instale el programa de manejador de mensajes en la región de CICS.

Resultados

La siguiente solicitud de servicio que se va a ejecutar a través de la interconexión de solicitante se procesa mediante el manejador de mensajes.

Qué hacer a continuación

Pruebe los cambios para la interconexión de solicitante para asegurarse de que las solicitudes de servicio se van a la ubicación correcta y que el programa de manejador de mensajes se está comportando como se diseñó.

Soporte para transacciones de servicios web

La especificación de Transacción Atómica de Servicios Web (o WS-AtomicTransaction) y la especificación Coordinación de Servicios Web (o WS-Coordination) definen protocolos para transacciones de corto plazo que permiten a los sistemas de proceso de transacciones interoperar en un entorno de servicios web. Las transacciones que utilizan WS-AtomicTransaction tienen las propiedades *ACID* de atomicidad, consistencia, aislamiento y durabilidad.

Las especificaciones se pueden encontrar en OASIS.

Nota: CICS soporta el nivel de las especificaciones de noviembre de 2004.

Las aplicaciones de CICS desplegadas como solicitantes o proveedores de servicios web pueden participar en transacciones distribuidas con otras implementaciones de servicio web que soportan las especificaciones.

Servicios de registro

Los Servicios de registro es la parte del modelo WS-Coordination que permite a la aplicación registrar protocolos de coordinación. En una transacción distribuida, los servicios de registro de los sistemas participantes se comunican uno con otro para permitir que las aplicaciones conectadas participen en esos protocolos.

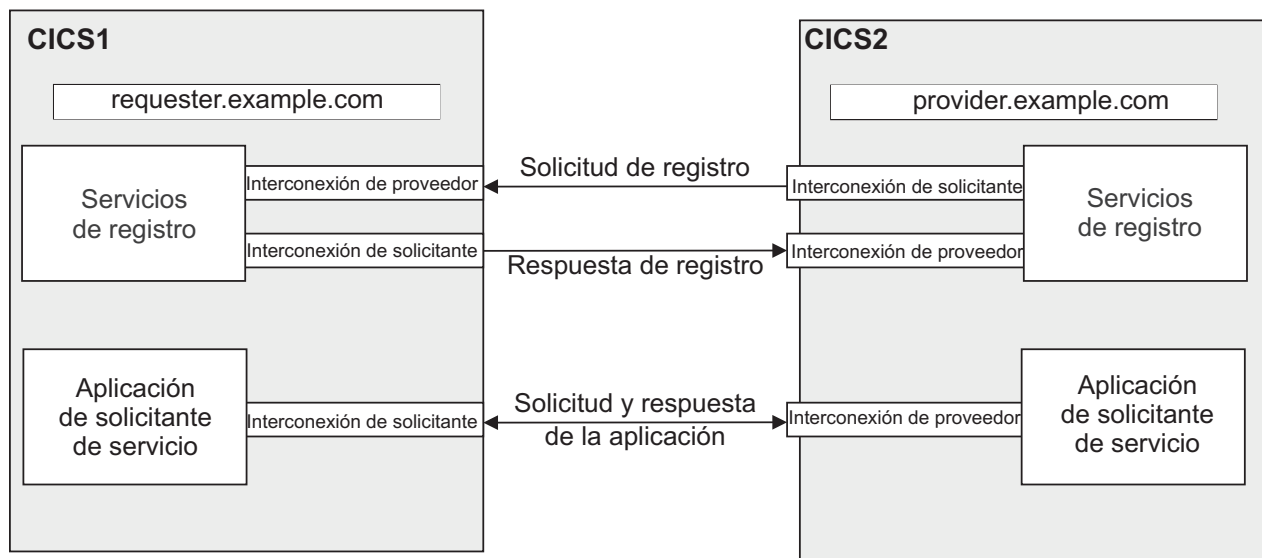


Figura 24. Servicios de registro

Figura 24 muestra dos sistemas CICS, CICS1 y CICS2. Una aplicación de solicitante de servicio en CICS1 invoca una aplicación de proveedor de servicios en CICS2. Las dos regiones de CICS y las aplicaciones se configuran para que las dos aplicaciones participen en una única transacción distribuida, utilizando los protocolos WS-Coordination. La aplicación de solicitante de servicio es el coordinador, y la aplicación de proveedor de servicios es el participante.

En favor de estos protocolos, los servicios de registro de las dos regiones de CICS interactúan al inicio de la transacción y de nuevo durante la finalización de la transacción. Durante estas interacciones, los servicios de registro de ambas regiones

pueden operar en distintos momentos como un proveedor de servicios y como un solicitante. Por lo tanto, en cada región, los servicios de registro utilizan una interconexión de proveedor de servicios y una interconexión de solicitante de servicio. Las interconexiones se definen para CICS con PIPELINE y los recursos asociados.

Los servicios de registro de cada región están asociados a una dirección de punto final. Por lo tanto, en este ejemplo, los servicios de registro de CICS1 tienen una dirección de punto final de requester.example.com; que en CICS2 tiene una dirección de punto final de provider.example.com.

En un CICSplex, puede distribuir la interconexión de proveedor de servicios de registro a una región diferente. Esto se muestra en la Figura 25.

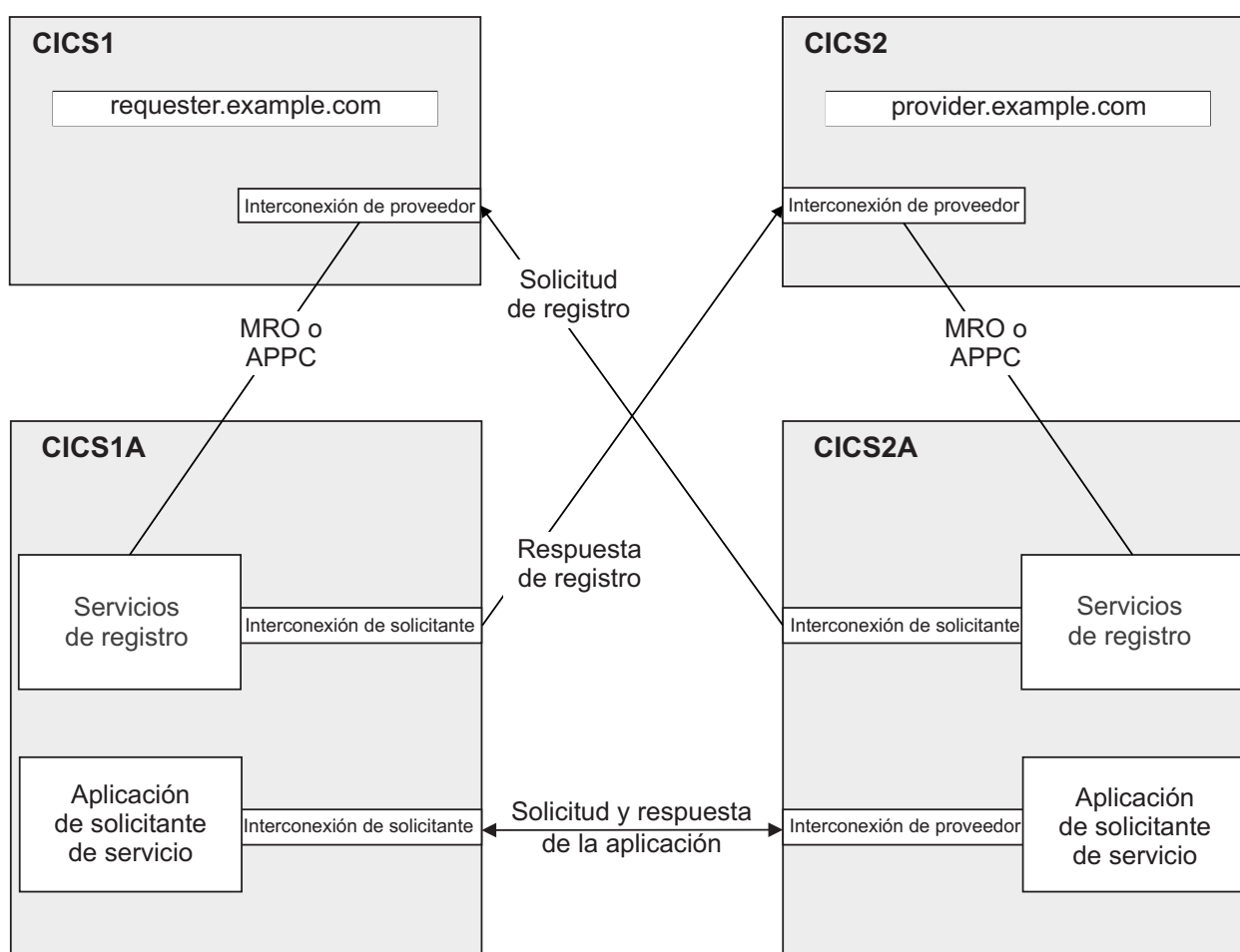


Figura 25. Servicios de registro en CICSplex

En esta configuración, la interconexión de proveedor se comunica con los servicios de registro utilizando MRO o APPC. La interconexión de solicitante de servicio de registro debe permanecer en la misma región que la interconexión de solicitante de la aplicación.

Esta configuración es útil cuando las aplicaciones de proveedor y solicitante de servicio se distribuyen entre un gran número de regiones. Cuando configura las interconexiones de la aplicación para participar en las transacciones de servicios web, debe proporcionar información sobre el punto final de servicios de registro proporcionando la dirección IP y el número de puerto de la interconexión de proveedor de servicios de registro. Teniendo un sólo punto final, puede simplificar la configuración porque todas las interconexiones contendrán la misma información. Por ejemplo, en Figura 25 en la página 194 la dirección IP que especifica en la interconexión de solicitante de la aplicación es `requester.example.com`.

Los mismos argumentos se aplican a la aplicación de proveedor de servicios. En el ejemplo, la interconexión de la aplicación de proveedor especificará una dirección IP de `requester.example.com`.

Configuración de CICS para transacciones de servicio web

Para que aplicaciones de proveedor y solicitante de servicio web participen en transacciones de servicio web, debe configurar CICS en consecuencia instalando varios recursos de CICS.

Antes de empezar

Antes de poder instalar estos recursos, debe conocer la ubicación de los archivos de configuración de interconexión que CICS proporciona en apoyo de las transacciones de servicio web. De forma predeterminada, los archivos de configuración se proporcionan en el directorio `/usr/lpp/cicsts/cicsts54/pipeline/configs`, pero la vía de acceso del archivo predeterminado puede haber cambiado durante la instalación de CICS.

Acerca de esta tarea

El soporte de CICS para transacciones de servicio web utiliza un servicio de registro proporcionado por CICS. Este servicio de registro consta de un proveedor de servicios y un solicitante de servicio. Debe instalar recursos para el proveedor de servicios y el solicitante de servicio; aunque las aplicaciones sean todas proveedores de servicios o solicitantes de servicio.

También debe instalar una definición de programa para el programa del manejador de cabeceras que se invoca cuando ejecuta las aplicaciones de proveedor y solicitante de servicios.

Los recursos que necesita para configurar CICS para transacciones de servicio web se proporcionan todas en el grupo DFHWSAT, excepto para DFHPIDIR que se proporciona en uno de los grupos siguientes: DFHPIVS, DFHPIVR o DFHPICF. El grupo DFHWSAT no se incluye en la lista DFHLIST y, por lo tanto, no se instala automáticamente. No puede cambiar los recursos proporcionados por CICS en el grupo DFHWSAT.

Para configurar CICS para transacciones de servicio web:

Procedimiento

1. Añada el conjunto de datos DFHPIDIR al JCL de inicio. DFHPIDIR almacena una correlación entre contextos y tareas.
 - a. Añada una sentencia de definición de datos nueva para el conjunto de datos DFHPIDIR en el JCL de inicio de CICS

- b. Cree el conjunto de datos DFHPIDIR utilizando la información de DFHDEFDS.JCL. El RECORDSIZE de DFHPIDIR predeterminado es 1 KB, que es adecuado para la mayoría de usos. Puede crear DFHPIDIR con un RECORDSIZE mayor si lo necesita.
- c. Instale el grupo adecuado para el conjunto de datos en el sistema CICS: DFHPIVS, DFHPIVR o DFHPICF. Para obtener más información sobre estos grupos, consulte Defining the WS-AT data set.

Si desea compartir el archivo DFHPIDIR en regiones de CICS, las regiones deben estar conectadas de forma lógica a través de MRO. Debe instalar un conjunto de datos por grupo de regiones que están actuando como un servidor lógico.

Consejo: Se recomienda no compartir conjuntos de datos entre regiones que no están conectadas lógicamente.

2. Copie el contenido del grupo DFHWSAT en otro grupo. No puede cambiar los recursos proporcionados por CICS en el grupo DFHWSAT. Sin embargo, debe cambiar el atributo CONFIGFILE en los recursos PIPELINE.
3. Modifique el recurso PIPELINE del proveedor de servicio de registro. La PIPELINE se denomina DFHWSATP, y especifica el archivo de configuración de interconexión /usr/lpp/cicsts/cicsts54/pipeline/configs/registrationservicePROV.xml en el atributo CONFIGFILE.
 - a. Cambie el atributo CONFIGFILE para reflejar la ubicación del archivo en el sistema.
 - b. Deje los demás atributos sin modificar.

Utilice el archivo de configuración de interconexión exactamente cómo se proporciona; no cambie el contenido.

4. Instale el recurso PIPELINE. El recurso PIPELINE del proveedor de servicios de registro no necesita estar en la misma región de CICS que las aplicaciones de proveedor o solicitante de servicio, pero debe estar conectados a esa región con una conexión MRO o APPC adecuada.
5. Sin cambiarlo, instale la URIMAP que ha usado el proveedor de servicios de registro en la misma región que PIPELINE. La URIMAP se denomina DFHRSURI.
6. Modifique el recurso PIPELINE del solicitante de servicio de registro. La PIPELINE se denomina DFHWSATR, y especifica el archivo de configuración de interconexión /usr/lpp/cicsts/cicsts54/pipeline/configs/registrationserviceREQ.xml en el atributo CONFIGFILE.
 - a. Cambie el atributo CONFIGFILE para reflejar la ubicación del archivo en el sistema.
 - b. Deje los demás atributos sin modificar.

Utilice el archivo de configuración de interconexión exactamente cómo se proporciona; no cambie el contenido.

7. Instale el recurso PIPELINE. El recurso PIPELINE del solicitante de servicios de registro debe estar en la misma región de CICS que las aplicaciones de proveedor y solicitante de servicio.
8. Instale los programas utilizados por la interconexión de proveedor de servicio de registro en la misma región que los recursos PIPELINE. Los programas son DFHWSATX, DFHWSATR y DFHPIRS. Si ambos recursos PIPELINE están en diferentes regiones, debe instalar estos programas en ambas regiones.

9. Instale la definición de recurso PROGRAM para el programa de manejador de cabeceras. El programa se denomina DFHWSATH. Instale el PROGRAM en las regiones donde se ejecutan las aplicaciones de solicitante y proveedor de servicios.

Resultados

CICS está configurado ahora para que las aplicaciones de proveedor o solicitante de servicio puedan participar en transacciones distribuidas utilizando los protocolos WS-AtomicTransaction y WS-Coordination.

Qué hacer a continuación

Debe configurar cada aplicación participante de manera individual.

Configuración de un proveedor de servicios para transacciones de servicios web

Si una aplicación de proveedor de servicios va a participar en transacciones de servicios web, el archivo de configuración de interconexión debe especificar un elemento `<headerprogram>` y un elemento `<service_parameter_list>`.

Antes de empezar

Si desea que su aplicación de proveedor de servicios participe en transacciones de servicios web, debe utilizar protocolos SOAP para comunicarse con el solicitante de servicio, y debe configurar su interconexión para utilizar uno de los manejadores de mensajes SOAP proporcionado por CICS. Aunque haya configurado la aplicación de proveedor de servicios correctamente, participará en transacciones de servicios web con el solicitante de servicio sólo si la aplicación de solicitante se ha configurado para participar.

Acerca de esta tarea

Además de la información de configuración de interconexión específica para la aplicación, el archivo de configuración debe contener información que utiliza CICS para garantizar que la aplicación participa en transacciones de servicios web.

CICS proporciona un ejemplo de archivo de configuración de interconexión que contiene esta información en el directorio de archivos `/usr/lpp/cicsts/cicsts54/samples/pipelines/wsatprovider.xml` (donde `/usr/lpp/cicsts/cicsts54` es el directorio de instalación predeterminado para archivos CICS en z/OS UNIX).

Procedimiento

1. En la definición del controlador de terminal, codifique un elemento `<headerprogram>` en el elemento `<cics_soap_1.1_handler>`, `<cics_soap_1.2_handler>`, `<cics_soap_1.1_handler_java>` o `<cics_soap_1.2_handler_java>`. Codifique los elementos `<program_name>`, `<namespace>`, `<localname>` y `<mandatory>` exactamente como se muestra en este ejemplo:

```
<terminal_handler>
  <cics_soap_1.1_handler>
    <headerprogram>
      <program_name>DFHWSATH</program_name>
      <namespace>http://schemas.xmlsoap.org/ws/2004/10/wscoor</namespace>
      <localname>CoordinationContext</localname>
```

```

        <mandatory>false</mandatory>
    </headerprogram>
</cics_soap_1.1_handler>
</terminal_handler>

```

Incluya otros elementos `<headerprogram>` si la aplicación los necesita.

2. Codifique un elemento `<registration_service_endpoint>` en `<service_parameter_list>`. Codifique `<registration_service_endpoint>` de la siguiente manera:

```

<registration_service_endpoint>
http://address:port/cicswsat/RegistrationService
</registration_service_endpoint>

```

address es la dirección IP de la región de CICS donde está ubicada la interconexión de proveedor de servicios de registro.

port es el número de puerto utilizado por la interconexión de proveedor de servicios de registro.

Codifique todo los demás exactamente como se muestra: la serie `cicswsat/RegistrationService` coincide con el atributo `PATH` de `URIMAP DFHRSURI`:

```

<registration_service_endpoint>
http://provider.example.com:7160/cicswsat/RegistrationService
</registration_service_endpoint>

```

Configuración de un solicitante de servicio para transacciones de servicio web

Si una aplicación de solicitante de servicio va a participar en transacciones de servicio web, el archivo de configuración de interconexión debe especificar un elemento `<headerprogram>` y un elemento `<service_parameter_list>`.

Antes de empezar

Si desea que su aplicación de solicitante de servicio participe en transacciones de servicios web, debe utilizar protocolos SOAP para comunicarse con el proveedor de servicios, y debe configurar su interconexión para utilizar uno de los manejadores de mensajes SOAP proporcionado por CICS. Aunque haya configurado la aplicación de solicitante de servicio correctamente, participará en transacciones de servicios web con el proveedor de servicios sólo si la aplicación de proveedor se ha configurado para participar.

Acerca de esta tarea

Además de la información de configuración de interconexión específica para la aplicación, el archivo de configuración debe contener información que utiliza CICS para garantizar que la aplicación participa en transacciones de servicios web.

CICS proporciona un ejemplo de archivo de configuración de interconexión que contiene esta información en el directorio de archivos `/usr/lpp/cicsts/cicsts54/samples/pipelines/wsatrequester.xml` (donde `/usr/lpp/cicsts/cicsts54` es el directorio de instalación predeterminado para archivos CICS en z/OS UNIX).

Procedimiento

1. Codifique un elemento `<headerprogram>` en el elemento `<cics_soap_1.1_handler>`, `<cics_soap_1.2_handler>`, `<cics_soap_1.1_handler_java>` o `<cics_soap_1.2_handler_java>`. Codifique los

elementos <program_name>, <namespace>, <localname> y <mandatory> exactamente como se muestra en el ejemplo siguiente:

```
<cics_soap_1.1_handler>
  <headerprogram>
    <program_name>DFHWSATH</program_name>
    <namespace>http://schemas.xmlsoap.org/ws/2004/10/wscoor</namespace>
    <localname>CoordinationContext</localname>
    <mandatory>true</mandatory>
  </headerprogram>
</cics_soap_1.1_handler>
```

Puede incluir otros elementos <headerprogram> si los necesita la aplicación.

2. Codifique un elemento <registration_service_endpoint> en <service_parameter_list>. Codifique <registration_service_endpoint> de la siguiente manera:

```
<registration_service_endpoint>
http://address:port/cicswsat/RegistrationService
</registration_service_endpoint>
```

address es la dirección IP de la región de CICS donde está ubicada la interconexión de proveedor de servicios de registro.

port es el número de puerto utilizado por la interconexión de proveedor de servicios de registro.

No debe haber espacio entre el inicio del elemento

<registration_service_endpoint>, su contenido y el final del elemento <registration_service_endpoint>. Se han incluido espacios en este ejemplo para mayor claridad.

3. Si desea que CICS cree un nuevo contexto transaccional para cada solicitud, en lugar de utilizar el mismo para solicitudes en la misma unidad de trabajo, añada el elemento vacío, <new_tx_context_required/>, de un <service_parameter_list> al archivo de configuración de interconexión:

```
<service_parameter_list>
  <registration_service_endpoint>
    http://requester.example.com:7159/cicswsat/RegistrationService
  </registration_service_endpoint>
  <new_tx_context_required/>
</service_parameter_list>
```

No debe haber espacio entre el inicio del elemento

<registration_service_endpoint>, su contenido y el final del elemento <registration_service_endpoint>. Se han incluido espacios en este ejemplo para mayor claridad.

El valor <new_tx_context_required/> no es el valor predeterminado para CICS, y no está incluido en el archivo de configuración de interconexión de ejemplo, wsatprovider.xml. Si añade el <new_tx_context_required/> de <service_parameter_list> al archivo de configuración de interconexión, se permiten las llamadas de bucle de retorno a CICS, por lo tanto, tenga en cuenta que se puede producir un punto muerto en esta situación.

Determinar si un mensaje SOAP es parte de una transacción atómica

Cuando se invoca un servicio web CICS en una interconexión de transacción atómica, el mensaje SOAP no tiene que ser parte de una transacción atómica necesariamente.

Acerca de esta tarea

El elemento <soapenv:Header> contiene información específica cuando el mensaje SOAP es parte de una transacción atómica. Para descubrir si el mensaje SOAP es parte de una transacción atómica, puede:

Procedimiento

- Mirar en el contenido del elemento <soapenv:Header> utilizando un rastreo.
 1. Llevar a cabo un rastreo adicional utilizando el componente PI y estableciendo el nivel de rastreo en 2.
 2. Buscar el punto de rastreo PI 0A31, que contiene la información para el contenedor de solicitud. En concreto, busque PIIS EVENT - REQUEST_CNT que aparece justo antes del elemento <cicswsa:Action>.
- Utilizar un programa de manejador de mensajes escrito por el usuario en las interconexiones DFHWSATP para visualizar el contenido del contenedor DFHREQUEST cuando contiene datos RECEIVE-REQUEST. Si opta por este enfoque, asegúrese de que define el programa de manejador de mensajes en el archivo de configuración de interconexión.

Ejemplo

El siguiente ejemplo muestra la información que puede ver en la cabecera del sobre SOAP para una transacción atómica.

```
<soapenv:Header>
  <wscoor:CoordinationContext soapenv:mustUnderstand="1"> 1
    <wscoor:Expires>500</wscoor:Expires>
    <wscoor:Identifier>com.ibm.ws.wstx:
      0000010a2b5008c80000000200000019a75aab901a1758a4e40e2731c61192a10ad6e921
    </wscoor:Identifier>
    <wscoor:CoordinationType>http://schemas.xmlsoap.org/ws/2004/10/wsat</wscoor:CoordinationType> 2
    <wscoor:RegistrationService 3
      xmlns:wscoor="http://schemas.xmlsoap.org/ws/2004/10/wscoor">
        <cicswsa:Address xmlns:cicswsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
          http://clientIPaddress:clientPort/_IBMSYSAPP/wscoor/services/RegistrationCoordinatorPort
        </cicswsa:Address>
        <cicswsa:ReferenceProperties
          xmlns:cicswsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
          <websphere-wsat:txID
            xmlns:websphere-wsat="http://wstx.Transaction.ws.ibm.com/extension">com.ibm.ws.wstx:
              0000010a2b5008c80000000200000019a75aab901a1758a4e40e2731c61192a10ad6e921
          </websphere-wsat:txID>
          <websphere-wsat:instanceID
            xmlns:websphere-wsat="http://wstx.Transaction.ws.ibm.com/extension">com.ibm.ws.wstx:
              0000010a2b5008c80000000200000019a75aab901a1758a4e40e2731c61192a10ad6e921
          </websphere-wsat:instanceID>
        </cicswsa:ReferenceProperties>
      </wscoor:RegistrationService>
    </wscoor:CoordinationContext>
  </soapenv:Header>
```

1. CoordinationContext indica que el mensaje SOAP está pensado para participar en una transacción atómica. Contiene la información necesaria para que el proveedor de servicios web sea parte del servicio de coordinación, suponiendo que el proveedor está configurado para reconocer y procesar la cabecera.
2. CoordinationType indica la versión de la especificación WS-AT que cumple con el contexto de coordinación.

3. La coordinación RegistrationService describe dónde está el punto de registro del coordinador y la información que debe devolver el servicio web participante al coordinador cuando intenta registrar un componente de la transacción atómica.

Comprobación del progreso de una transacción atómica

Cuando se invoca un servicio web CICS como parte de una transacción atómica, la transacción pasa a través de varios estados. Estos estados indican si la transacción ha sido correcta o ha tenido que retrotraerse.

Acerca de esta tarea

Si necesita acceder a esta información, puede:

Procedimiento

- Mire el contenido del elemento <cicswsa:Action> utilizando el rastreo.
 1. Lleve a cabo un rastreo adicional utilizando el componente PI y estableciendo el nivel de rastreo en 2.
 2. Busque el punto de rastreo PI 0A31, que contiene la información para el contenedor de solicitud. En concreto, busque PIIS EVENT - REQUEST_CNT que aparece justo antes del elemento <cicswsa:Action>.
- Utilice un programa de manejador de mensajes escrito por el usuario en las interconexiones DFHWSATR y DFHWSATP para visualizar el contenido de los contenedores DFHWS-SOAPACTION. Si opta por este enfoque, asegúrese de que define el programa de manejador de mensajes en los archivos de configuración de interconexión.

Ejemplo

Los estados para una transacción que se completa correctamente y que se confirma son:

```
"http://schemas.xmlsoap.org/ws/2004/10/wscoor/Register"  
"http://schemas.xmlsoap.org/ws/2004/10/wscoor/RegisterResponse"  
"http://schemas.xmlsoap.org/ws/2004/10/wsac/Prepare"  
"http://schemas.xmlsoap.org/ws/2004/10/wsac/Prepared"  
"http://schemas.xmlsoap.org/ws/2004/10/wsac/Commit"  
"http://schemas.xmlsoap.org/ws/2004/10/wsac/Committed "
```

Los estados para una transacción que se retrotrae son:

```
"http://schemas.xmlsoap.org/ws/2004/10/wscoor/Register"  
"http://schemas.xmlsoap.org/ws/2004/10/wscoor/RegisterResponse"  
"http://schemas.xmlsoap.org/ws/2004/10/wsac/Rollback"  
"http://schemas.xmlsoap.org/ws/2004/10/wsac/Aborted"
```

Soporte para la optimización MTOM/XOP de datos binarios

En los mensajes SOAP estándar, los objetos binarios están codificados en base 64 e incluidos en el cuerpo del mensaje, lo que aumenta su tamaño un 33%. En el caso de objetos binarios de gran tamaño, este aumento de tamaño puede afectar significativamente al tiempo de transmisión. La implementación de MTOM/XOP proporciona una solución a este problema.

Las especificaciones del Mecanismo de optimización de transmisión de mensajes SOAP (MTOM) y de Empaquetado optimizado binario XML (XOP), a menudo conocidas como MTOM/XOP, definen un método para la optimización de la transmisión de objetos de datos base64Binary grandes dentro de mensajes SOAP.

- La especificación MTOM define conceptualmente un método para optimizar mensajes SOAP separando datos binarios, que de otra manera estarían codificados en base64, y enviarlo en archivos adjuntos binarios separados utilizando un mensaje MIME Multipart/Related. Este tipo de mensaje MIME se denomina *Mensaje MTOM*. El envío de datos en formato binario reduce de forma significativa su tamaño, optimizando de esta forma la transmisión del mensaje SOAP.
- La especificación XOP define una implementación para optimizar mensajes XML utilizando archivos de datos adjuntos binarios en un formato de empaquetado que incluye, pero no está limitado a, mensajes MIME.

CICS implementa soporte para estas especificaciones en interconexiones de solicitante y proveedor cuando el protocolo de transporte es WebSphere MQ, HTTP o HTTPS. Como alternativa para incluir datos base64Binary directamente en el mensaje SOAP, las aplicaciones CICS que se despliegan como solicitantes o proveedores de servicios web pueden utilizar este soporte para enviar y recibir mensajes MTOM con archivos adjuntos binarios.

Puede configurar este soporte utilizando opciones adicionales en el archivo de configuración de interconexiones.

MTOM/XOP y SOAP

Cuando se utiliza MTOM/XOP para optimizar un mensaje SOAP, se serializa en un mensaje MIME Multipart/Related utilizando un proceso XOP. Los datos base64Binary se extraen del mensaje SOAP y se empaquetan como archivos adjuntos binarios individuales dentro del mensaje MIME, de forma similar a los archivos adjuntos de correo electrónico.

El tamaño de los datos base64Binary se reduce significativamente porque los archivos adjuntos están codificados en formato binario. El XML en el mensaje SOAP se convierte en formato XOP sustituyendo los datos base64Binary por un elemento `<xop:Include>` especial que hace referencia al archivo adjunto MIME relevante utilizando el URI.

El mensaje SOAP modificado se denomina *documento XOP*, y forma el documento raíz dentro del mensaje. El documento XOP y los archivos adjuntos binarios forman el *paquete XOP*. Cuando se aplica a la especificación SOAP MTOM, el paquete XOP es un mensaje MIME en formato MTOM.

El documento raíz se identifica haciendo referencia a su ID-contenido en la cabecera de tipo de contenido global del mensaje MIME. Aquí tiene un ejemplo de una cabecera de tipo de contenido:

```
Content-Type: Multipart/Related; boundary=MIME_boundary;
  type="application/soap+xml"; start="<claim@insurance.com>"
```

El parámetro **start** contiene el ID-contenido del documento XOP. Si este parámetro no se incluye en la cabecera de tipo de contenido, la primera parte del mensaje MIME se supone que es el documento XOP.

El orden de los archivos adjuntos en el mensaje MIME no es importante. En algunos mensajes, por ejemplo, los archivos adjuntos binarios pueden aparecer antes del documento XOP. Una aplicación que maneja los mensajes MIME no debe confiar en los archivos adjuntos que aparecen en un orden específico. Para obtener información detallada, lea las especificaciones MTOM/XOP.

El ejemplo siguiente demuestra cómo un sólo mensaje SOAP que contiene una imagen JPEG se optimiza utilizando el proceso XOP. El mensaje SOAP es de la siguiente manera:

```
<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xmime="http://www.w3.org/2003/12/xop/mime">
  <soap:Body>
    <submitClaim>
      <accountNumber>5XJ45-3B2</accountNumber>
      <eventType>accident</eventType>
      <image xmime:contentType="image/jpeg" xsi:type="base64binary">4f3e..(encoded image)</image>
    </submitClaim>
  </soap:Body>
</soap:Envelope>
```

Una versión MTOM/XOP de este mensaje SOAP es de la siguiente manera:

```
MIME-Version: 1.0
Content-Type: multipart/related; boundary=MIME_boundary;
  type="application/soap+xml"; start="<claim@insurance.com>" 1

--MIME_boundary
Content-Type: application/soap+xml; charset=UTF-8
Content-Transfer-Encoding: 8bit
Content-ID: <claim@insurance.com> 2

<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:xop='http://www.w3.org/2004/08/xop/include'
  xmlns:xop-mime='http://www.w3.org/2005/05/xmlmime'>
  <soap:Body>
    <submitClaim>
      <accountNumber>5XJ45-3B2</accountNumber>
      <eventType>accident</eventType>
      <image xop-mime:content-type='image/jpeg'><xop:Include href="cid:image@insurance.com"/></image> 3
    </submitClaim>
  </soap:Body>
</soap:Envelope>

--MIME_boundary
Content-Type: image/jpeg
Content-Transfer-Encoding: binary
Content-ID: <image@insurance.com> 4

...binary JPG image...

--MIME_boundary--
```

1. El parámetro **start** indica qué parte del mensaje MIME es el documento XOP raíz.
2. El valor ID-contenido identifica una parte del mensaje MIME. En este caso, es el documento XOP raíz.
3. El elemento `<xop:Include>` hace referencia al archivo adjunto binario JPEG.
4. El ID-contenido identifica el JPEG en el archivo adjunto binario.

Mensajes MTOM y archivos de datos adjuntos binarios en CICS

CICS soporta y controla el manejo de mensajes MTOM en las interconexiones de solicitante y proveedor de servicios web utilizando un programa de manejador MTOM y un proceso XOP.

Configure y habilite el soporte MTOM utilizando el archivo de configuración de interconexión. Si el soporte MTOM está habilitado para una interconexión, CICS desempaqueta los mensajes MTOM de entrada automáticamente y empaqueta los mensajes de salida. Si el soporte MTOM no está habilitado para una interconexión y CICS recibe un mensaje MTOM, las interconexiones basadas en Java aceptan el mensaje MTOM de entrada; sin embargo, tras interconexiones SOAP rechazan el mensaje MTOM de entrada con un error SOAP.

Opciones de configuración para interconexiones basadas en Java

Puede configurar una interconexión de proveedor para realizar las siguientes tareas:

- Aceptar mensajes MTOM, pero nunca enviar mensajes de respuesta MTOM.
- Aceptar mensajes MTOM y siempre enviar mensajes de respuesta MTOM.
- Procesar documentos XOP y archivos adjuntos binarios en modalidad Axis2.

Puede configurar una interconexión de solicitante para realizar las siguientes tareas:

- No enviar nunca un mensaje MTOM, pero aceptar mensajes de respuesta MTOM.
- Enviar siempre mensajes MTOM y aceptar mensajes de respuesta MTOM.
- Procesar documentos XOP y archivos adjuntos binarios en modalidad Axis2.

Opciones de configuración para interconexiones que no soportan Java

Puede configurar una interconexión de proveedor para realizar las siguientes tareas:

- Aceptar mensajes MTOM, pero nunca enviar mensajes de respuesta MTOM.
- Aceptar mensajes MTOM y enviar el mismo tipo de mensaje de respuesta.
- Aceptar mensajes MTOM, pero sólo enviar mensajes MTOM cuando haya archivos adjuntos binarios presentes.
- Aceptar mensajes MTOM y siempre enviar mensajes de respuesta MTOM.
- Procesar documentos XOP y archivos adjuntos binarios en modalidad directa o de compatibilidad.

Puede configurar una interconexión de solicitante para realizar las siguientes tareas:

- No enviar nunca un mensaje MTOM, pero aceptar mensajes de respuesta MTOM.
- Sólo enviar mensajes MTOM cuando haya archivos adjuntos binarios y aceptar mensajes de respuesta MTOM.
- Enviar siempre mensajes MTOM y aceptar mensajes de respuesta MTOM.
- Procesar documentos XOP y archivos adjuntos binarios en modalidad directa o de compatibilidad.

Modalidades de soporte

Hay tres modalidades de soporte proporcionadas en la interconexión para gestionar documentos XOP y archivos adjuntos binarios asociados.

Modalidad Axis2

La modalidad Axis2 se utiliza cuando el controlador de terminal de la interconexión de servicios web es el manejador de mensajes `<cics_soap_1.1_handler_java>` o `<cics_soap_1.2_handler_java>`.

Modalidad directa

En modalidad directa, los archivos binarios asociados con un mensaje MTOM de entrada o de salida se pasan en contenedores a través de la interconexión y son manejados directamente por la aplicación, sin necesidad de realizar ninguna conversión de datos.

Modalidad de compatibilidad

La modalidad de compatibilidad se utiliza cuando el proceso de la interconexión requiere que el mensaje esté en formato XML estándar, con los datos binarios almacenados como campos base64Binary dentro del mensaje. Para mensajes de entrada, el documento XOP y los archivos adjuntos binarios se reorganizan en un mensaje XML estándar, al comienzo de la interconexión cuando está habilitada la seguridad de servicios web, o al final de la interconexión, cuando está habilitada la validación de servicios web. Para mensajes de salida, se crea un mensaje XML estándar y se pasa a través de la interconexión. Este mensaje XML se convierte a formato XOP mediante el manejador MTOM justo antes de que CICS lo envíe.

La modalidad de compatibilidad es mucho menos eficaz que la modalidad directa porque los datos binarios se convierten a formato en base64 y luego se vuelven a convertir al formato binario. Sin embargo, no permite a los servicios web interoperar con otros proveedores o solicitantes de servicios web MTOM/XOP sin necesidad de cambiar las aplicaciones.

Proceso de mensajes MTOM de entrada para interconexiones que no soportan Java:

Cuando se habilita el manejador MTOM en interconexiones que no soportan Java, comprueba las cabeceras del mensaje de entrada en el contenedor DFHREQUEST o DFHRESPONSE para determinar el formato del mensaje durante el proceso de manejo del transporte.

Cuando se recibe un mensaje MIME Multipart/Related, el manejador MTOM desempaqueta el mensaje de la siguiente manera:

1. Coloca las cabeceras y datos binarios de cada archivo adjunto binario en contenedores individuales.
2. Coloca una lista de contenedores en el contenedor DFHWS-XOP-IN.
3. Coloca el documento XOP, que ha formado la raíz del mensaje, de vuelta en el contenedor DFHREQUEST o DFHRESPONSE, sustituyendo el mensaje original.

Si no hay archivos adjuntos binarios, el documento XOP se gestiona como un mensaje XML normal y no se necesita un proceso XOP. Si hay archivos adjuntos binarios, se habilita el proceso XOP para el mensaje.

Si el proceso XOP está habilitado, el manejador MTOM comprueba las propiedades de interconexión para determinar si el mensaje actual debe procesarse en modalidad directa o de compatibilidad, y coloca esta información en el contenedor DFHWS-MTOM-IN.

En modalidad de proveedor, el manejador MTOM también crea el contenedor DFHWS-MTOM-OUT para determinar cómo se debe procesar el mensaje de respuesta de salida.

Modalidad directa

Cuando está utilizando el soporte de servicios web CICS, es decir, cuando una interconexión de proveedor de servicios utiliza el manejador de aplicación DFHPITP o se invoca una interconexión de solicitante de servicio mediante **INVOKE WEBSERVICE**, la interconexión puede procesar el documento XOP y archivos adjuntos binarios en modalidad directa.

En esta modalidad, el manejador MTOM pasa el documento XOP y los contenedores asociados al siguiente manejador de mensajes en la interconexión para su proceso. El soporte de servicios web CICS interpreta los elementos <xop:Include>. Si el campo base64Binary se representa como un contenedor en la estructura de datos de aplicación, el nombre de contenedor del archivo adjunto se almacena en la estructura. Si el campo se representa como una serie de longitud fija o variable, el contenido del contenedor se copia en el campo de la estructura de datos de la aplicación relevante. La estructura de datos pasa al programa de aplicación.

Modalidad de compatibilidad

Si la interconexión está configurada para utilizar un manejador de aplicación personalizado, o Web Services Security también está habilitado, el mensaje se procesa en modalidad de compatibilidad. En esta modalidad, el documento XOP y los archivos adjuntos binarios se vuelven a constituir inmediatamente en un mensaje SOAP utilizando el proceso XOP, de forma que el contenido se procesa correctamente en la interconexión. El proceso XOP lleva a cabo las siguientes tareas:

1. Explora el documento XOP para elementos <xop:Include>, sustituyendo cada aparición con los datos binarios del archivo adjunto de referencia en formato codificado en base64.
2. Descarta el contenedor DFHWS-XOP-IN y todos los contenedores de archivo adjunto.

El mensaje SOAP reconstituido se pasa al siguiente manejador en la interconexión para procesarse de forma normal.

Si se habilita la validación del servicio web, la interconexión conmuta a la modalidad de compatibilidad cuando el mensaje alcanza el manejador de aplicación. El mensaje se vuelve a constituir en un mensaje SOAP validado y pasado a la aplicación.

Proceso de mensajes MTOM de salida para interconexiones que no soportan Java:

Cuando una interconexión que no soporta Java se configura para enviar mensajes MTOM de salida, se comprueban las propiedades de interconexión y servicio web para determinar cómo se debe procesar y enviar el mensaje.

Estas propiedades están almacenadas en dos contenedores, DFHWS-MTOM-OUT y DFHWS-XOP-OUT. En una interconexión de modalidad de solicitante, CICS crea estos contenedores cuando la aplicación emite el mandato **EXEC CICS INVOKE WEBSERVICE**. En una interconexión de modalidad de proveedor, el contenedor DFHWS-MTOM-OUT ya se ha inicializado con las opciones que se determinaron cuando se recibió el mensaje de entrada.

Si el mensaje de salida se puede procesar en modalidad directa, la optimización del mensaje tiene lugar inmediatamente. Si el mensaje de salida tiene que procesarse en modalidad de compatibilidad, la optimización tiene lugar al final del proceso de interconexión.

Si no ha desplegado la aplicación de solicitante o proveedor de servicios web utilizando el asistente de servicios web de CICS, o si ha habilitado la validación de servicios web o la Seguridad de servicios web en la interconexión, el mensaje de salida se procesa en modalidad de compatibilidad.

Modalidad directa

En modalidad directa, tiene lugar el siguiente proceso:

1. Se construye un documento XOP desde la estructura de datos de la aplicación en el contenedor DFHWS-DATA. Se identifica cualquier campo binario igual o mayor en tamaño de 1500 bytes, y los datos binarios y cabeceras MIME que describen el archivo adjunto binario se colocan en contenedores separados. Si los datos binarios ya están en un contenedor, ese contenedor se utiliza directamente como archivo adjunto. A continuación, se inserta un elemento `<xop:Include>` XML en lugar de los datos binarios codificados con base64 habituales utilizando un ID-contenido generado. Por ejemplo:

```
<xop:Include href="cid:generated-content-ID-value"
xmlns:xop="http://www.w3.org/2004/08/xop/include">
```
2. Todos los contenedores se añaden a la lista de archivos adjuntos en el contenedor DFHWS-XOP-OUT.
3. Cuando el manejador SOAP ha procesado DFHWS-DATA, el documento XOP y el sobre SOAP se almacenan en el contenedor DFHREQUEST o DFHRESPONSE y se procesan a través de la interconexión.
4. Cuando finaliza el último manejador de mensajes, el manejador MTOM empaqueta el documento XOP y los archivos adjuntos binarios en el mensaje MIME Multipart/Related y lo envía al proveedor o solicitante de servicio web. El contenedor DFHWS-XOP-OUT y cualquier contenedor asociado se descartan.

Modalidad de compatibilidad

Si la interconexión no es capaz de manejar el documento XOP directamente, tiene lugar el siguiente proceso:

1. El cuerpo SOAP se construye en DFHWS-DATA desde la estructura de datos de la aplicación y se procesa en la interconexión como normal.
2. Cuando el manejador final ha terminado de procesar el mensaje, el manejador MTOM comprueba las opciones en el contenedor DFHWS-MTOM-OUT para determinar si se debe utilizar MTOM, teniendo en cuenta, opcionalmente, si están presentes los archivos adjuntos binarios. Si el manejador MTOM determina que no se necesita MTOM, no tiene lugar ningún proceso XOP y CICS no envía ningún mensaje SOAP como hace habitualmente.
3. Si el manejador MTOM determina que se debe enviar un mensaje de salida en formato MTOM, el proceso XOP explora el mensaje para ver qué campos son elegibles para dividir los datos en archivos adjuntos binarios. Para que un campo sea elegible, debe tener el atributo de MIME **contentType** especificado en el elemento y el valor binario asociado debe constar de datos base64Binary válidos en formato canónico. El tamaño de los datos debe ser mayor de 1500 bytes. El proceso XOP crea los archivos adjuntos binarios y la lista de archivos adjuntos y, a continuación, sustituye los campos por elementos `<xop:Include>`.
4. El manejador MTOM empaqueta el documento XOP y los archivos adjuntos binarios como mensaje MIME Multipart/Related y CICS lo envía al proveedor o solicitante de servicio web.

Restricciones al utilizar MTOM/XOP

Para soportar MTOM/XOP, puede especificar el elemento `<mtom>` en el archivo de configuración de interconexión o habilitar el manejador MTOM en la interconexión. Sin embargo, hay restricciones asociadas a cada método.

Restricciones para interconexiones basadas en Java:

Especificando el elemento <mtom> en un archivo de configuración de interconexión habilita el soporte MTOM/XOP para la interconexión basada en Java. Sin embargo, hay restricciones con esta implementación de MTOM/XOP.

Manejador de aplicación DFHPITP

La modalidad Axis2 del soporte MTOM/XOP no se puede utilizar con interconexiones que especifican DFHPITP como el manejador de aplicación.

Seguridad de servicios web

La modalidad Axis2 del soporte MTOM/XOP no se puede utilizar con interconexiones que utilizan configuraciones de seguridad de servicios web que requieren firmas XML.

Uso del mandato INQUIRE PIPELINE

Si se emite un mandato **INQUIRE PIPELINE** en una interconexión basada en Java utilizando la modalidad Axis2 del soporte MTOM/XOP, los atributos **Mtomst**, **Sendmtomst**, **Mtomnoxopst**, **Xopsupportst** y **Xopdirectst** se notifican como Nomtom. Para obtener más información, consulte INQUIRE PIPELINE.

Restricciones para otras interconexiones SOAP:

Habilitar el manejador MTOM en la interconexión significa que puede soportar implementaciones de servicio web que utilizan la optimización MTOM/XOP. La opción de modalidad de compatibilidad significa que puede interoperar con estos servicios web sin necesidad de cambiar las aplicaciones de servicios web. Sin embargo, hay determinadas situaciones donde no puede utilizar MTOM/XOP o su uso está restringido.

Uso del asistente de servicios web de CICS

La optimización de modalidad directa para MTOM/XOP sólo está disponible si está utilizando DFHWS2LS en un nivel de correlación de al menos 1.2, y el documento WSDL contiene al menos un campo de tipo xsd:base64Binary. Los servicios web habilitados que utilizan DFHLS2WS no son elegibles para la optimización XOP.

Los servicios web generados utilizando DFHLS2WS con CHAR-VARYING=BINARY especificado pueden ser elegibles para las optimizaciones MTOM/XOP. Otros servicios web generados utilizando DFHLS2WS no contienen datos binarios y no son elegibles para las optimizaciones MTOM/XOP, pero funcionarán normalmente en una PIPELINE que soporta MTOM/XOP.

Interconexiones de proveedor

CICS proporciona un manejador de aplicación predeterminado llamado DFHPITP que se puede configurar en una interconexión de proveedor. Este manejador de aplicación es capaz de manejar documentos XOP y de crear los contenedores necesarios para soportar el proceso de interconexión en modalidad directa y de compatibilidad. Si está utilizando su propio manejador de aplicación en una interconexión de proveedor y desea habilitar MTOM/XOP, debe configurar la interconexión para que se ejecute en modalidad de compatibilidad.

Interconexiones de solicitante

Si sus aplicaciones utilizan el mandato **INVOKE WEBSERVICE**, CICS gestiona la optimización del mensaje SOAP en modalidad directa y de compatibilidad. Si está utilizando el programa DFHPITP para iniciar la interconexión, sólo puede enviar y recibir mensajes MIME Multipart/Related en modalidad de compatibilidad.

Seguridad de servicios web

Si habilita el manejador MTOM en el archivo de configuración de interconexión para que se ejecute directamente en modalidad directa, y también puede habilitar el manejador de mensajes de la Seguridad de servicios web, la interconexión sólo soporta la gestión de mensajes MTOM en modalidad de compatibilidad.

Manejo de datos binarios

Cuando tiene muchos datos binarios a incluir en el servicio web, por ejemplo, un archivo gráfico como un JPEG, puede utilizar MTOM/XOP para optimizar el tamaño del mensaje que se envía al solicitante o proveedor de servicios. El tamaño mínimo de datos binarios se pueden optimizar utilizando MTOM/XOP es 1500 bytes. Si los datos binarios de un campo son menos de 1500 bytes, CICS no optimiza el campo.

Como se indica en la especificación XOP, no debe haber espacios en blanco en los datos base64Binary. Los programas de aplicación que producen datos base64Binary deben utilizar el formato canónico. Si los datos base64Binary de un mensaje de salida no contienen espacios en blanco, CICS no convierte los datos en un archivo adjunto binario. Cuando CICS genera datos base64Binary, los campos se proporcionan en formato canónico y, por lo tanto, no contienen espacios en blanco.

El atributo **contentType** debe estar presente en campos base64Binary para que el proceso XOP tenga lugar en modalidad de compatibilidad en mensajes de salida. El atributo **contentType** no debe estar presente en campos hexBinary.

Validación del servicio web

Si activa la validación del servicio web, tiene lugar el siguiente proceso de interconexión:

- Si se ha pasado un documento XOP de entrada a través de la interconexión en modalidad directa, CICS cambia automáticamente a la modalidad de compatibilidad y lo convierte de nuevo en XML estándar cuando el soporte de servicio web de CICS va a validar el documento.
- Se genera un mensaje SOAP de salida como XML estándar y se procesa en modalidad de compatibilidad.

Se necesita el proceso de interconexión adicional porque el proceso de validación no puede gestionar el contenido de los documentos XOP.

Configuración de CICS para soportar MTOM/XOP

Para soportar mensajes MTOM en CICS, debe especificar el soporte MTOM/XOP correcto para el tipo de interconexión en los archivos de configuración de interconexión.

Configuración del soporte MTOM/XOP para interconexiones basadas en Java:

Para configurar el soporte MTOM/XOP para interconexiones basadas en Java, debe añadir el elemento <mtom> a los archivos de configuración de interconexión.

Antes de empezar

Antes de realizar esta tarea, debe identificar o crear los archivos de configuración de interconexión a los que añadirá la información de configuración para MTOM/XOP.

Acerca de esta tarea

Si se define el elemento <mtom> en el archivo de configuración de interconexión, el soporte MTOM se habilita para todos los mensajes de entrada y de salida. Sin embargo, si este elemento no se especifica en el archivo de configuración de interconexión, el soporte MTOM se habilita únicamente para mensajes de entrada.

Procedimiento

Añada un elemento <mtom> al archivo de configuración de interconexión. Este elemento debe definirse después del elemento <addressing> opcional y antes del elemento <headerprogram> opcional.

Ejemplo

Para una interconexión modalidad solicitante o proveedor, puede especificar:

```
<cics_soap_1.2_handler_java>
  <jvmserver>JVMSESV1</jvmserver>
  <addressing></addressing>
  <mtom></mtom>
  <headerprogram>
    <program_name>HDRPROG4</program_name>
    <namespace>http://mynamespace</namespace>
    <localname>myheaderblock</localname>
    <mandatory>true</mandatory>
  </headerprogram>
</cics_soap_1.2_handler_java>
```

Configuración de MTOM/XOP para otras interconexiones SOAP:

Para configurar el soporte MTOM/XOP para interconexiones que no utilizan manejadores <cics_soap_1.1_handler_java> o <cics_soap_1.2_handler_java>, debe añadir el manejador MTOM a los archivos de configuración de interconexión.

Antes de empezar

Antes de realizar esta tarea, debe identificar o crear los archivos de configuración de interconexión a los que añadirá la información de configuración para MTOM/XOP.

Procedimiento

1. Añada un elemento <cics_mtom_handler> al archivo de configuración de interconexión. Este elemento debe ser el primero en el elemento <provider_pipeline>, y el último elemento antes de <service_parameter_list> en el elemento <requester_pipeline>. Codifique los siguientes elementos:

```
<cics_mtom_handler>
  <dfhmtom_configuration version="1">
  </dfhmtom_configuration>
</cics_mtom_handler>
```

El elemento <dfhmtom_configuration> es un contenedor para los otros elementos de la configuración. Si desea aceptar los valores predeterminados para el proceso MTOM/XOP, puede especificar un elemento vacío de la siguiente manera:

```
<cics_mtom_handler/>
```

2. Opcional: Codifique un elemento `<mtom_options>`. En la interconexión de solicitante de servicio como del proveedor de servicios, este elemento especifica si el mensaje de salida debe estar empaquetado como un mensaje MTOM.
 - a. Codifique el atributo **send_mtom** para definir si el mensaje de salida debe enviarse como mensaje MTOM. Para obtener detalles de este atributo, consulte “Elemento `<mtom_options>`” en la página 118.
 - b. Codifique el atributo **send_when_no_xop** para definir si el mensaje de salida debe enviarse como un mensaje MTOM cuando no hay archivos adjuntos binarios presentes. Para obtener detalles de este atributo, consulte “Elemento `<mtom_options>`” en la página 118.
3. Opcional: Codifique un elemento `<xop_options>` con un atributo **apphandler_supports_xop**. Esto especifica si el manejador de aplicación es capaz de manejar documentos XOP directamente. Si no incluye este atributo, el valor predeterminado depende de si el elemento `<apphandler>` especifica DFHPITP u otro programa. Para obtener detalles de este atributo, consulte “Elemento `<xop_options>`” en la página 119.
4. Opcional: Codifique un elemento `<mime_options>` con un atributo **content_id_domain**. Este especifica el nombre de dominio que debe utilizarse cuando se generan valores ID-contenido de MIME, que se utilizan para identificar archivos adjuntos binarios. Para obtener detalles de este atributo, consulte “Elemento `<mime_options>`” en la página 121.

Ejemplo

El ejemplo siguiente muestra un elemento `<cics_mtom_handler>` completado en el que están presentes elementos opcionales:

```
<provider_pipeline>
  <cics_mtom_handler>
    <dfhmtom_configuration version="1">
      <mtom_options send_mtom="same" send_when_no_xop="no" />
      <xop_options apphandler_supports_xop="yes" />
      <mime_options content_id_domain="example.org" />
    </dfhmtom_configuration>
  </cics_mtom_handler>
  ....
</provider_pipeline>
```

Soporte para Web Services Addressing

Los servicios de soportar de CICS que utilizan las especificaciones de Web Services Addressing (WS-Addressing) del Worldwide Web Consortium (W3C). Esta familia de especificaciones proporciona mecanismos independientes al transporte para direccionar servicios web y facilitar el direccionamiento de extremo a extremo.

CICS garantiza que las aplicaciones de servicio web existentes pueden aceptar solicitudes de servicios web que utilizan WS-Addressing. También puede crear nuevos servicios web que utilicen propiedades de referencias de punto final y de direccionamiento de mensajes en mensajes SOAP.

WS-Addressing añade información de direccionamiento, en forma de MAP (Message Addressing Properties - Propiedades de direccionamiento de mensajes), a las cabeceras de mensajes SOAP. Las MAP incluyen información de mensajería, por ejemplo un ID de mensaje exclusivo y referencias de punto final que detallan de dónde viene el mensaje, a dónde va el mensaje y a qué lugar se deben enviar los mensajes de respuesta o de anomalía. Una referencia de punto final (EPR) es un

tipo específico de MAP que incluye la dirección de destino del mensaje, parámetros de referencia opcionales para que los use la aplicación y metadatos opcionales.

Características del soporte de WS-Addressing

CICS incluye las siguientes características para dar soporte a WS-Addressing:

- Las aplicaciones de solicitante y proveedor de servicios pueden interactuar con otros servicios que está utilizando WS-Addressing sin necesidad de volver a desplegarlos. Un nuevo manejador de mensajes, el manejador de mensajes de direccionamiento DFHWSADH, en la interconexión direcciona mensajes que contienen información de WS-Addressing para el servicio web especificado.
- Se puede escribir una aplicación que utilice mandatos de API de WS-Addressing para crear una referencia de punto final y para crear, actualizar, suprimir y consultar un contexto de direccionamiento.
- Puede dirigir los mensajes de respuesta a puntos finales distintos al punto final del solicitante; por ejemplo, se pueden dirigir los mensajes de error a un manejador de errores dedicado.
- Se pueden pasar parámetro de referencia a aplicaciones como parte de las MAP en la cabecera SOAP.

Soporte para especificaciones e interoperatividad de WS-Addressing

De forma predeterminada, CICS soporta las especificaciones recomendadas:

- W3C WS-Addressing 1.0 - Clase Core
- Enlace W3C WS-Addressing 1.0 - SOAP
- W3C WS-Addressing 1.0 - Metadatos

Estas especificaciones las identifica el espacio de nombres <http://www.w3.org/2005/08/addressing>. Si no se indica algo distinto, la semántica de WS-Addressing que se describe en este documento hace referencia a las especificaciones recomendadas.

Para interoperatividad, CICS también soporta la especificación de envío:

- W3C WS-Addressing- Submission

Esta especificación la identifica el espacio de nombres <http://schemas.xmlsoap.org/ws/2004/08/addressing>. Utilice la especificación de envío sólo si debe interoperar con un cliente o proveedor de servicios web que implementa la especificación de envío .

Descripción general de Web Services Addressing

Web Services Addressing (WS-Addressing) proporciona un marco de trabajo estándar para especificar los puntos finales de un mensaje SOAP. Es un marco de trabajo neutral para el transporte y mejora la interoperabilidad de los servicios web que utilizan diferentes mecanismos de transporte. La especificación de WS-Addressing introduce referencias de punto final y propiedades de direccionamiento de mensajes.

Web Services Addressing (WS-Addressing) es una especificación de Worldwide Web Consortium (W3C) que mejora la interoperabilidad entre los servicios web definiendo una forma estándar de direccionar servicios web y de proporcionar información de direccionamiento en mensajes SOAP. Los mensajes SOAP se

pueden enviar a través de varios mecanismos de transporte, incluido HTTP y WebSphere MQ, cada uno de los cuales almacena la información de destino para el mensaje de forma diferente.

Los servicios web de CICS existentes que se despliegan en una interconexión configurada para utilizar WS-Addressing pueden utilizar los valores de WS-Addressing predeterminados sin necesidad de cambios. Para sacar partido de las prestaciones de WS-Addressing, utilice los mandatos de API de WS-Addressing. La implementación de WS-Addressing soporta un error SOAP para cada operación WSDL.

Propiedades de direccionamiento de mensajes

Las propiedades de direccionamiento de mensajes (MAP) son un conjunto de las propiedades WS-Addressing bien definidas que se pueden representar como elementos en cabeceras SOAP. Las MAP proporcionan una manera estándar de transmitir información, como el punto final al que se debe dirigir el mensaje, o la información sobre la relación que tiene el mensaje con otros mensajes. Las MAP definidas por la especificación de WS-Addressing se resumen en la siguiente tabla.

Tabla 11. Propiedades de direccionamiento de mensaje definidas por la especificación WS-Addressing

Nombre de MAP de WS-Addressing abstracta	Nombre de MAP de WS-Addressing SOAP	Tipo de contenido MAP	Multiplicidad	Descripción
[action]	<wsa:Action>	xs:anyURI	1..1	Un URI absoluto que identifica de forma exclusiva la semántica del mensaje. Este valor es obligatorio.
[destination]	<wsa:To>	xs:anyURI en el mensaje SOAP EndpointReference en el contexto de direccionamiento	0..1	El URI absoluto que especifica la dirección del destinatario deseado del mensaje. Si no se especifica el valor, se utiliza el valor predeterminado para el URI anónimo definido en la especificación: http://www.w3.org/2005/08/addressing/anonymous . En el contexto de direccionamiento, la MAP <wsa:To> se representa como EPR. Cuando se envía <wsa:To> como parte de un mensaje SOAP, se divide en su dirección y sus parámetros de referencia, como define el esquema.
[reference parameters] *	[reference parameters]*	xs:any	0..unbounded	Los parámetros que se corresponden con las propiedades <wsa:ReferenceParameters> de la referencia de punto final a la que se dirige el mensaje. Este valor es opcional.
[source endpoint]	<wsa:From>	EndpointReference	0..1	Referencia al punto final desde el que se origina el mensaje. Este valor es opcional.
[reply endpoint]	<wsa:ReplyTo>	EndpointReference	0..1	Una referencia de punto final para el receptor planificado de las respuestas a este mensaje. Este valor es opcional. Si no se especifica el valor, va siempre a http://www.w3.org/2005/08/addressing/anonymous .
[fault endpoint]	<wsa:FaultTo>	EndpointReference	0..1	Referencia de punto final para el receptor al que van dirigidos los errores relacionados con este mensaje. Este valor es opcional y predeterminado para el valor de la MAP <wsa:ReplyTo>.

Tabla 11. Propiedades de direccionamiento de mensaje definidas por la especificación WS-Addressing (continuación)

Nombre de MAP de WS-Addressing abstracta	Nombre de MAP de WS-Addressing SOAP	Tipo de contenido MAP	Multiplicidad	Descripción
[relationship] *	<wsa:RelatesTo>	xs:anyURI más atributo opcional de tipo xs:anyURI	0..unbounded	Un par de valores que indica cómo se relaciona este mensaje con otro mensaje. El contenido de este elemento transmite el <wsa:MessageID> del mensaje relacionado. Un atributo opcional revela el tipo de relación. Este valor es opcional. Si no se especifica el valor, va siempre a http://www.w3.org/2005/08/addressing/reply .
[message id]	<wsa:MessageID>	xs:anyURI		Un URI absoluto que identifica de forma exclusiva el mensaje. Este valor es opcional; si no se proporciona, CICS genera un valor para las solicitudes y respuestas de entrada.

El siguiente es un ejemplo de un mensaje SOAP que contiene las MAP de WS-Addressing:

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsa="http://w3.org/2005/08/addressing"
  xmlns:example="http://example.ibm.com/namespace">
  <S:Header>
    ...
    <wsa:To>http://example.ibm.com/enquiry</wsa:To>
    <wsa:ReplyTo>
      <wsa:Address>http://example.ibm.com/enquiryReply</wsa:Address>
    </wsa:ReplyTo>
    <wsa:Action>...</wsa:Action>
    <example:AccountCode wsa:IsReferenceParameter='true'>123456789</example:AccountCode>
    <example:DiscountId wsa:IsReferenceParameter='true'>ABCDEF</example:DiscountId>
    ...
  </S:Header>
  <S:Body>
    ...
  </S:Body>
</S:Envelope>
```

Referencias de punto final

Una referencia de punto final es tipo específico de MAP, que proporciona un mecanismo estándar para encapsular información sobre puntos finales específicos. Las referencias de punto final se pueden enviar a otras partes y se pueden utilizar como destino del punto final de servicio web que representan. La siguiente tabla resume el modelo de información para las referencias de punto final.

Tabla 12. Modelo de información para las referencias de punto final

Nombre de propiedad abstracto	Tipo de propiedad	Multiplicidad	Descripción
[address]	xs:anyURI	1..1	El URI absoluto que especifica la dirección del punto final.

Tabla 12. Modelo de información para las referencias de punto final (continuación)

Nombre de propiedad abstracto	Tipo de propiedad	Multiplidad	Descripción
[reference parameters] *	xs:any	0..unbounded	Los elementos de información del elemento cualificado del espacio de nombres son necesarios para interactuar con el punto final.
[metadata]	xs:any	0..unbounded	Descripción del comportamiento, políticas y prestaciones del punto final.

El siguiente fragmento de XML ilustra una referencia de punto final. El elemento `<wsa:EndpointReference>` hace referencia al punto final en el URI `http://example.ibm.com/enquiry` y contiene metadatos que especifican la interfaz a la que hace referencia la referencia de punto final y algunos parámetros de referencia específicos de la aplicación.

```
<wsa:EndpointReference
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
  xmlns:example="http://example.ibm.com/namespace">
  <wsa:Address>http://example.ibm.com/enquiry</wsa:Address>
  <wsa:Metadata
    xmlns:wsdl="http://www.w3.org/ns/wsdl-instance"
    wsdl:wsdlLocation="http://example.ibm.com/wsdl/wsdl-location.wsdl">
    <wsam:InterfaceName>example:reservationInterface</wsam:InterfaceName>
  </wsa:Metadata>
  <wsa:ReferenceParameters>
    <example:AccountCode>123456789</example:AccountCode>
    <example:DiscountId>ABCDEFGH</example:DiscountId>
  </wsa:ReferenceParameters>
</wsa:EndpointReference>
```

Las MAP de WS-Addressing de tipo `wsa:EndpointReferenceType` son: `<wsa:From>`, `<wsa:ReplyTo>` y `<wsa:FaultTo>`. Sin embargo, la MAP `<wsa:To>` se define en el WS-Addressing 1.0 estándar como que tiene un tipo de `xs:anyURI`. Para mayor simplicidad, CICS trata las MAP `<wsa:To>` en el contexto de direccionamiento como EPRs. Cuando se envía una `<wsa:To>` como parte de un mensaje SOAP, CICS lo divide en su dirección y sus parámetros de referencia, como requiere el estándar.

Espacios de nombres predeterminados

Se hace referencia al siguiente prefijo y los espacios de nombres correspondientes a través de la documentación de WS-Addressing:

Tabla 13. Prefijo y espacio de nombres correspondiente

Prefijo predeterminado	Espacio de nombres
xs	<code>http://www.w3.org/2001/XMLSchema</code>
wsa	<code>http://www.w3.org/2005/08/addressing</code> (esquema de recomendación) <code>http://schemas.xmlsoap.org/ws/2004/08/addressing</code> (esquema de envío)
wsam	<code>http://www.w3.org/2007/05/addressing/metadata</code>

Configuración de una interconexión de solicitante para Web Services Addressing

Para configurar una interconexión de solicitante para soportar Web Services Addressing (WS-Addressing), debe añadir un manejador de direccionamiento al archivo de configuración de interconexión.

Antes de empezar

Debe identificar o crear el archivo de configuración de interconexión para añadir la información de configuración para WS-Addressing. También debe decidir qué especificaciones de WS-Addressing utilizar. Utilice la especificación *W3C WS-Addressing 1.0 Core* donde sea posible.

Acerca de esta tarea

Puede añadir soporte para WS-Addressing de una de estas dos formas:

- Si la interconexión SOAP utiliza Java, Axis2 maneja el proceso SOAP y puede utilizar el soporte proporcionado por esta tecnología para gestionar solicitudes que utilizan WS-Addressing. Todo el manejo de cabeceras lo gestiona Axis2, y es importante que no añada el programa de proceso de cabeceras DFHWSADH a la interconexión. Puede utilizar sus propios programas de proceso de cabeceras. Para obtener un mejor rendimiento, escriba manejadores Axis2 en Java si desea procesar cabeceras SOAP.
- Si la interconexión SOAP no utiliza Java, debe añadir el programa de proceso de cabeceras proporcionado por CICS, DFHWSADH, para manejar las solicitudes.

Procedimiento

- Si la interconexión SOAP utiliza un elemento `<cics_soap_1.1_handler_java>` o `<cics_soap_1.2_handler_java>`, añada un elemento `<addressing>` al archivo de configuración de interconexión. Incluya un elemento `<namespace>` que contiene la especificación que desea utilizar en el mensaje de solicitud, que puede ser diferentes al mensaje de respuesta; por ejemplo, siempre puede enviar una solicitud que cumpla con la especificación principal W3C, aunque el mensaje de respuesta utilice la especificación de envío. Axis2 soporta ambas especificaciones de WS-Addressing en mensajes de entrada.

El ejemplo siguiente muestra cómo puede configurar la interconexión de solicitante:

```
<requester_pipeline>
  <service>
    <service_handler_list>
      <cics_soap_1.1_handler_java>
        <jvmserver>JVMSEV1</jvmserver>
        <addressing>
          <namespace>http://www.w3.org/2005/08/addressing</namespace>
        </addressing>
      </cics_soap_1.1_handler_java>
    </service_handler_list>
  </service>
</requester_pipeline>
```

El elemento `<jvmserver>` contiene el nombre del recurso JVMSEV1 que soporta Axis2.

- Si la interconexión SOAP no utiliza Java, añada un programa de cabecera de direccionamiento CICS en `<cics_soap_1.1_handler>` o `<cics_soap_1.2_handler>` para el archivo de configuración de interconexión. El ejemplo siguiente muestra cómo puede configurar la interconexión de solicitante:


```

<requester_pipeline>
  <service>
    <service_handler_list>
      <cics_soap_1.1_handler>
        <headerprogram>
          <program_name>DFHWSADH</program_name>
          <namespace>http://www.w3.org/2005/08/addressing</namespace>
          <localname>*</localname>
          <mandatory>true</mandatory>
        </headerprogram>
      </cics_soap_1.1_handler>
    </service_handler_list>
  </service>
</requester_pipeline>

```

Codifique los elementos <program_name>, <localname> y <mandatory> exactamente como se muestra. Establezca <namespace> en <http://www.w3.org/2005/08/addressing> para utilizar la especificación *W3C WS-Addressing 1.0 Core* o <http://schemas.xmlsoap.org/ws/2004/08/addressing> para utilizar la especificación *W3C WS-Addressing Submission*.

El orden de los programas de proceso de cabeceras no se garantiza. Si define otros programas de proceso de cabeceras, añádalos a un elemento del manejador SOAP de CICS posterior en el elemento <service_handler_list>. El manejador de cabecera DFHWSADH debe estar en el primer elemento de manejador SOAP.

Resultados

La interconexión de solicitante está configurada para soportar WS-Addressing.

Qué hacer a continuación

Cree un recurso PIPELINE que señale al archivo de configuración. Si está utilizando una interconexión SOAP basada en Java, asegúrese de que hay un recurso JVMSERVER habilitado para manejar el proceso Axis2.

Configuración de una interconexión de proveedor para Web Services Addressing

Para configurar una interconexión de proveedor para soportar Web Services Addressing (WS-Addressing), debe añadir un manejador de direccionamiento al archivo de configuración de interconexión.

Antes de empezar

Debe identificar o crear el archivo de configuración de interconexión para añadir la información de configuración para WS-Addressing. También debe decidir qué especificaciones de WS-Addressing utilizar. Utilice la especificación *W3C WS-Addressing 1.0 Core* donde sea posible.

Acerca de esta tarea

Puede añadir soporte para WS-Addressing de una de estas dos formas:

- Si la interconexión SOAP utiliza Java, Axis2 maneja el proceso SOAP y puede utilizar el soporte proporcionado por esta tecnología para gestionar solicitudes que utilizan WS-Addressing. Todo el manejo de cabeceras lo gestiona Axis2, y es importante que no añada el programa de proceso de cabeceras DFHWSADH a la interconexión. Puede utilizar sus propios programas de proceso de cabeceras. Para obtener un mejor rendimiento, escriba manejadores Axis2 en Java si desea procesar cabeceras SOAP.

- Si la interconexión SOAP no utiliza Java, debe añadir el programa de proceso de cabeceras proporcionado por CICS, DFHWSADH, para manejar las solicitudes.

Procedimiento

- Si la interconexión SOAP utiliza un elemento `<cics_soap_1.1_handler_java>` o `<cics_soap_1.2_handler_java>`, añada un elemento `<addressing>` al archivo de configuración de interconexión. Opcionalmente, puede incluir uno o más elementos `<namespace>`. Este elemento contiene la especificación que desea utilizar en el mensaje de salida, que puede ser diferente al mensaje de entrada; por ejemplo, siempre puede enviar una respuesta de salida que cumpla con la especificación W3C, aunque el mensaje de entrada utilice la especificación de envío. Si excluye este elemento, Axis2 utiliza la misma especificación en el mensaje de salida como mensaje de entrada. Axis2 soporta ambas especificaciones de WS-Addressing en mensajes de entrada.

El ejemplo siguiente muestra cómo puede configurar la interconexión de proveedor:

```
<provider_pipeline>
  <terminal_handler>
    <cics_soap_1.1_handler_java>
      <jvmserver>JVMSEVR1</jvmserver>
      <addressing>
        <namespace>http://www.w3.org/2005/08/addressing</namespace>
      </addressing>
    </cics_soap_1.1_handler_java>
  </terminal_handler>
</provider_pipeline>
```

El elemento `<jvmserver>` contiene el nombre del recurso JVMSEVR que soporta Axis2.

- Si la interconexión SOAP no utiliza Java, añada el programa de cabecera de direccionamiento CICS, DFHWSADH, al manejador SOAP en el archivo de configuración de interconexión. El ejemplo siguiente muestra cómo puede configurar la interconexión de proveedor:

```
<provider_pipeline>
  <terminal_handler>
    <cics_soap_1.1_handler>
      <headerprogram>
        <program_name>DFHWSADH</program_name>
        <namespace>http://www.w3.org/2005/08/addressing</namespace>
        <localname>*</localname>
        <mandatory>true</mandatory>
      </headerprogram>
    </cics_soap_1.1_handler>
  </terminal_handler>
</provider_pipeline>
```

Codifique los elementos `<program_name>`, `<localname>` y `<mandatory>` exactamente como se muestra. Establezca `<namespace>` en `http://www.w3.org/2005/08/addressing` para utilizar la especificación W3C *WS-Addressing 1.0 Core* o `http://schemas.xmlsoap.org/ws/2004/08/addressing` para utilizar la especificación W3C *WS-Addressing Submission*.

El orden de los programas de proceso de cabeceras no se garantiza. Si define otros programas de proceso de cabeceras, añádalos en otro elemento de manejador SOAP de CICS en un elemento `<service_handler_list>`. El manejador de cabecera DFHWSADH debe estar en el último elemento de manejador SOAP.

Resultados

La interconexión de proveedor está configurada para soportar WS-Addressing.

Qué hacer a continuación

Cree un recurso PIPELINE que señale al archivo de configuración. Si está utilizando una interconexión SOAP basada en Java, asegúrese de que hay un recurso JVMSERVER habilitado para manejar el proceso Axis2.

Creación de un servicio web que utiliza WS-Addressing

Para crear un servicio web desde un documento WSDL que utiliza Web Services Addressing (WS-Addressing), utilice parámetros en el asistente de servicios web para manejar la conversión de XML a estructuras de lenguaje.

Acerca de esta tarea

Puede utilizar el trabajo del asistente de servicios web, DFHWS2LS, para controlar cómo se gestiona una referencia de punto final (EPR) en un documento WSDL y para determinar si CICS construye las acciones de entrada, salida y de error.

Procedimiento

1. Establezca el parámetro **MINIMUM-RUNTIME** en el asistente de servicios web, DFHWS2LS, en 3.0 o superior. Un nivel de tiempo de ejecución de al menos 3.0 garantiza que cualquier enlace de servicio web generado soporta completamente WS-Addressing y puede interoperar con otras plataformas de servicios web.
2. Establezca el parámetro **MAPPING-LEVEL** en el asistente de servicios web, DFHWS2LS, en 3.0 o superior.
3. Establezca el parámetro **WSADDR-EPR-ANY** en TRUE si desea utilizar los elementos de tipo `wsa:EndpointReferenceType` en los mensajes de solicitud o respuesta. Las referencias de punto final se pueden incluir en los datos de aplicación y tiene la opción de utilizar EPR en los mandatos de API como **WSACONTEXT BUILD**. Establecer el parámetro **WSADDR-EPR-ANY** en TRUE indica que CICS no debe transformar el EPR en una estructura de lenguaje en el tiempo de ejecución; en su lugar, CICS trata los datos EPR como un elemento `<xsd:any>` y lo almacena en un contenedor con nombre.

Este fragmento WSDL de ejemplo muestra un MAP `<wsa:To>` que se está pasando como elemento de tipo `wsa:EndpointReferenceType`:

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="exampleEPR" targetNamespace="http://example.ibm.com/"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:s0="http://example.ibm.com/"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata">
  <types>
    <xs:schema targetNamespace="http://test.org/"
      xmlns:s="http://www.w3.org/2001/XMLSchema"
      xmlns:s0="http://example.ibm.com/"
      xmlns:wsa="http://www.w3.org/2005/08/addressing">
      ...
      <xs:element name="exampleResponse" type="s0:typeResponse"/>
      <xs:complexType name="typeResponse">
        <xs:sequence>
          <xs:element name="myEpr" type="wsa:EndpointReferenceType"/> 1
        </xs:sequence>
      </xs:complexType>
    </xs:schema>
  </types>
</definitions>
```

```

...
</xs:schema>
</types>
...
<message name="msgResponse">
  <part element="s0:exampleResponse" name="response"/>
</message>
...
</definitions>

```

Cuando DFHWS2LS procesa el elemento `<xs:element name="myEpr" type="wsa:EndpointReferenceType"/>` **1** con el parámetro **WSADDR-EPR-ANY** establecido en TRUE, los datos del elemento myEpr se almacenan en un contenedor con nombre como un elemento `<xsd:any>` y un puntero al contenedor añadido a la estructura de lenguaje generada.

Por ejemplo, la estructura de lenguaje COBOL generada por DFHWS2LS para el elemento myEpr se muestra aquí:

```

09 myEpr.
   12 myEpr-xml-cont          PIC X(16).
   12 myEpr-xmlns-cont        PIC X(16).

```

El contenedor myEpr-xml-cont almacena el nombre del contenedor que posee los datos myEpr. myEpr-xmlns-cont es un contenedor opcional que se llena con declaraciones de espacio de nombres XML que están en el ámbito.

4. Guarde y envíe el trabajo DFHWS2LS.

Resultados

CICS crea un enlace de servicios web para manejar la transformación de datos y las estructuras de lenguaje que puede utilizar para crear la aplicación de proveedor de servicios o de solicitante de servicio.

Qué hacer a continuación

Para habilitar el servicio web, realice una exploración de interconexión para crear los recursos CICS necesarios.

Referencias de punto final predeterminadas:

La mayoría de los documentos WSDL contienen la dirección en la que está alojado el servicio web. En WS-Addressing, el documento WSDL también puede contener una referencia de punto final (EPR) para el servicio web. Esta EPR puede contener metadatos adicionales para facilitar la comunicación entre las aplicaciones de solicitante y proveedor.

Si utiliza DFHWS2LS para procesar el WSDL, la EPR se guarda en el enlace de servicio web y CICS la utiliza para enviar mensajes de solicitud y respuesta. Los parámetros de referencia, `<wsa:ReferenceParameters>`, que se definen en la EPR se incluyen en el mensaje SOAP. Esta EPR es conocida como la EPR predeterminada, porque puede sustituirse por la aplicación. Si la aplicación no proporciona una EPR explícita, se utiliza la EPR predeterminada de WSDL.

El siguiente fragmento de WSDL 1.1 incluye una EPR predeterminada:
`<soap:address location="http://example.ibm.com:12345/exampleTest" />`. El elemento `<port>` incluye un elemento hijo, `<wsa:EndpointReference>`, la dirección especificada por el elemento hijo, **2**, debe coincidir con la dirección especificada por el elemento padre, **1**:

```

<service name="exampleService">
  <port name="examplePort" binding="s0:createBinding">
    <soap:address location="http://example.ibm.com:12345/exampleTest" /> 1

    <wsa:EndpointReference
      xmlns:example="http://example.ibm.com/namespace"
      xmlns:wsdl="http://www.w3.org/2006/01/wsdl-instance"
      wsdl:wsdlLocation="http://example.ibm.com/location "
      title="http://example.ibm.com/example/example_wsdl"
      class="http://example.ibm.com/example/example_wsdl">
        <wsa:Address>http://example.ibm.com:12345/exampleTest</wsa:Address> 2
        <wsa:Metadata>
          <wsam:InterfaceName>example:Inventory</wsam:InterfaceName>
        </wsa:Metadata>
        <wsa:ReferenceParameters>
          <example:AccountCode>123456789</example:AccountCode>
          <example:DiscountId>ABCDEFG</example:DiscountId>
        </wsa:ReferenceParameters>
      </wsa:EndpointReference>
    </port>
  </service>

```

Acciones explícitas:

Los documentos WSDL pueden definir explícitamente los valores de las propiedades <wsa:Action>. Si el documento WSDL no contiene las propiedades <wsa:Action> definidas explícitamente, CICS compila acciones predeterminadas cuando DFHWS2LS procesa el WSDL.

WSDL 1.1

El siguiente fragmento WSDL 1.1 representa un sistema de registro que contiene las propiedades <wsa:Action> definidas explícitamente:

```

<definitions targetNamespace="http://example.ibm.com/namespace" ...>
  ...
  <portType name="bookingSystem">
    <operation name="makeBooking">
      <input message="tns:makeBooking"
        wsa:Action="http://example.ibm.com/namespace/makeBooking"
      </input>
      <output message="tns:bookingResponse"
        wsa:Action="http://example.ibm.com/namespace/bookingResponse"
      </output>
    </operation>
  </portType>
  ...
</definitions>

```

En este ejemplo, la acción de entrada de la operación makeBooking se define explícitamente como http://example.ibm.com/namespace/makeBooking, y la acción de salida se define explícitamente como http://example.ibm.com/namespace/bookingResponse.

WSDL 2.0

El siguiente fragmento WSDL 2.0 representa un sistema de registro que contiene las propiedades <wsa:Action> definidas explícitamente:

```

<description targetNamespace="http://example.ibm.com/namespace" ...>
  ...
  <interface name="bookingInterface">
    <operation name="makeBooking" pattern="http://www.w3.org/ns/wsdl/in-out">
      <input element="tns:makeBooking" messageLabel="In"
        wsa:Action="http://example.ibm.com/namespace/makeBooking"/>
    </operation>
  </interface>

```

```

        <output element="tns:makeBookingResponse" messageLabel="Out"
            wsa:Action="http://example.ibm.com/namespace/makeBookingResponse"/>
    </operation>
</interface>
...
</description>

```

En este ejemplo, la acción de entrada de la operación makeBooking se define explícitamente como http://example.ibm.com/namespace/makeBooking, y la acción de salida se define como http://example.ibm.com/namespace/makeBookingResponse.

Para obtener más información, consulte la especificación W3C WS-Addressing 1.0 Metadata.

Acciones predeterminadas para WSDL 1.1:

Si un documento WSDL 1.1 no contiene las propiedades <wsa:Action> especificadas explícitamente, CICS compila las acciones de entrada, salida y error predeterminadas cuando DFHWS2LS procesa el WSDL.

Acciones de entrada y salida predeterminadas para WSDL 1.1

CICS utiliza el siguiente patrón, en documentos WSDL 1.1 que siguen el esquema de recomendación o el esquema de envío para una acción de entrada o salida predeterminada:

```
[target namespace]/[port type name]/[input|output name]
```

Acciones de error predeterminadas para WSDL 1.1

Si está siguiendo el esquema de recomendación, la forma en la que CICS compila la acción de error predeterminada difiere del comportamiento descrito en el esquema. CICS, utiliza el siguiente patrón, en documentos WSDL 1.1 que siguen el esquema de recomendación, para construir un mensaje de error predeterminado. Tenga en cuenta que el nombre de error se omite.

```
[target namespace]/[port type name]/[operation name]/Fault/
```

Si está siguiendo el esquema de envío, la forma en la que CICS compila la acción de error predeterminada sigue el comportamiento descrito en el esquema. CICS utiliza el siguiente patrón, en documentos WSDL 1.1 que siguen el esquema de envío, para construir un mensaje de error predeterminado:

```
[target namespace]/[port type name]/[operation name]/Fault/[fault name]
```

Ejemplo de acciones predeterminadas generadas por CICS para un documento WSDL 1.1

Este ejemplo de sistema de registro ilustra cómo CICS construye acciones predeterminadas desde un documento WSDL 1.1:

```

<description targetNamespace="http://example.ibm.com/namespace" ...>
    ...
    <portType name="bookingInterface">
        <operation name="makeBooking">
            <input element="tns:makeBooking" name="MakeBooking"/>
            <output element="tns:bookingResponse" name="BookingResponse"/>
            <fault message="tns:InvalidBooking" name="InvalidBooking"/>
        </operation>
    </interface>
    ...
</definitions>

```

El fragmento de WSDL tiene las siguientes propiedades de direccionamiento:

Nombre de propiedad	Valor
[targetNamespace]	http://example.ibm.com/namespace
[portType name]	bookingInterface
[operation name]	makeBooking
[input name]	MakeBooking
[output name]	BookingResponse
[fault name]	InvalidBooking

Las siguientes acciones se crean a partir de estos valores:

Acción	Valor
Entrada de acción	http://example.ibm.com/namespace/bookingInterface/MakeBooking Si no se especifica [input name], se utiliza el valor de [operation name] con "Solicitud" añadido. Por ejemplo, en este caso Acción de entrada es http://example.ibm.com/namespace/bookingInterface/makeBookingRequest. http://example.ibm.com/namespace/bookingInterface/BookingResponse
Acción de salida	Si no se especifica [output name], se utiliza el valor de [operation name] con "Respuesta" añadido. Por ejemplo, en este caso la Acción de salida es http://example.ibm.com/namespace/bookingInterface/makeBookingResponse
Acción de error (Esquema de recomendación)	http://example.ibm.com/namespace/bookingInterface/MakeBooking/Fault/ Tenga en cuenta que [fault name] se omite.
Acción de error (Esquema de envío)	http://example.ibm.com/namespace/bookingInterface/MakeBooking/Fault/ InvalidBooking

Para obtener más información, consulte la especificación W3C WS-Addressing 1.0 Metadata.

Acciones predeterminadas para WSDL 2.0:

Si un documento WSDL 2.0 no contiene las propiedades <wsa:Action> especificadas explícitamente, CICS compila las acciones de entrada, salida y error predeterminadas cuando DFHWS2LS procesa el WSDL.

Acciones de entrada y salida predeterminadas para WSDL 2.0

CICS utiliza el siguiente patrón, en documentos WSDL 2.0 que siguen el esquema de recomendación, para construir acciones predeterminadas para entradas y salidas:

[target namespace]/[interface name]/[operation name][direction token]

Acciones de error predeterminadas para WSDL 2.0

Si está siguiendo el esquema de recomendación, la forma en la que CICS compila la acción predeterminada para WS-Addressing difiere del comportamiento descrito

en el esquema. Si está siguiendo el esquema de envío, la forma en la que CICS compila la acción predeterminada para errores de WS-Addressing sigue el comportamiento descrito en el esquema.

CICS utiliza el siguiente patrón, en documentos WSDL 2.0 que siguen el esquema de recomendación, para construir una acción predeterminada para errores: Tenga en cuenta que el nombre de error se omite.

[target namespace]/[interface name]/

CICS utiliza el siguiente patrón, en documentos WSDL 2.0 que siguen el esquema de envío, para construir una acción predeterminada para errores:

[target namespace]/[interface name]/[fault name]

Ejemplo de acciones predeterminadas generadas por CICS para un documento WSDL 2.0

Este ejemplo muestra cómo CICS construye acciones predeterminadas para un documento WSDL 2.0 siguiendo el esquema de recomendación:

```
<description targetNamespace="http://example.ibm.com/namespace" ...>
...
<interface name="bookingInterface">
  <operation name="makeBooking" pattern="http://www.w3.org/ns/wsd1/in-out">
    <input element="tns:makeBooking" messageLabel="In"/>
    <output element="tns:bookingResponse" messageLabel="Out"/>
  </operation>
</interface>
...
</definitions>
```

El fragmento de WSDL tiene las siguientes propiedades de direccionamiento:

Nombre de propiedad	Valor
[targetNamespace]	http://example.ibm.com/namespace
[interface name]	bookingInterface
[operation name]	makeBooking
[direction token]	Puede ser Request o Response.

Se crean las siguientes acciones de entrada y salida desde estos valores:

Acción	Valor
Entrada de acción	http://example.ibm.com/namespace/bookingInterface/makeBookingRequest
Acción de salida	http://example.ibm.com/namespace/bookingInterface/makeBookingResponse

Para obtener más información, consulte la especificación W3C WS-Addressing 1.0 Metadata.

Intercambios de mensajes

Web Services Addressing (WS-Addressing) soporta estos intercambios de mensajes: unidireccional, solicitud-respuesta bidireccional, solicitud-respuesta síncronas y solicitud-respuesta asíncronas.

Los intercambios de mensajes de Web Services Addressing implican las propiedades de direccionamiento de mensajes (MAP) y las referencias de punto final (EPR).

En tiempo de ejecución, CICS garantiza que la cabecera SOAP del mensaje de solicitud contiene la información de mensaje de WS-Addressing relevante, la aplicación de solicitante no tiene que establecer las cabeceras de WS-Addressing y, es posible, que no sepan que está utilizando WS-Addressing.

Unidireccional

Este sencillo mensaje unidireccional se define como una operación de sólo entrada. El Web Services Description Language (WSDL) para esta operación toma la forma siguiente:

```
<operation name="myOperation">
  <input message="tns:myInputMessage"/>
</operation>
```

Si está utilizando WS-Addressing, CICS añade las MAP <wsa:Action> y la MAP <wsa:MessageID> a la cabecera de mensaje SOAP del mensaje de solicitud de WS-Addressing en el tiempo de ejecución para garantizar la conformidad con la especificación de WS-Addressing.

La MAP <wsa:MessageID> es un ID exclusivo, si no se especifica, CICS genera este ID automáticamente.

Las MAP <wsa:Action> se derivan del WSDL y se almacenan en el archivo WSBind.

Puede sustituir los valores de estas MAP utilizando los mandatos de API de CICS WS-Addressing.

Solicitud-respuesta bidireccional

Este intercambio bidireccional implica un mensaje de solicitud y un mensaje de respuesta. La parte de la respuesta de la operación se puede definir como un mensaje de salida, un mensaje de error o ambos. La definición WSDL para una operación solicitud-respuesta toma la forma siguiente:

```
<operation name="myOperation">
  <input message="tns:myInputMessage"/>
  <output message="tns:myOutputMessage"/>
  <fault="tns:myFaultMessage"/>
</operation>
```

Las respuestas a, o los errores generados desde, solicitudes dirigidas a puntos finales se dirigen a la MAP <wsa:ReplyTo> o a la MAP <wsa:FaultTo> dependiendo de si el tipo de respuesta es normal o un error.

Especifique una MAP <wsa:ReplyTo> o <wsa:FaultTo> en el mensaje de solicitud para indicar dónde se debe enviar la respuesta.

Si está utilizando las especificaciones de recomendación y no especifica un valor para la MAP <wsa:ReplyTo>, la MAP <wsa:ReplyTo> va siempre a la referencia de punto final que contiene el URO anónimo (<http://www.w3.org/2005/08/addressing/anonymous>), que hace que CICS envíe la respuesta de vuelta al solicitante.

Si está utilizando las especificaciones de recomendación y no especifica un valor para la MAP <wsa:FaultTo>, la MAP <wsa:FaultTo> utiliza el valor predeterminado de la MAP <wsa:ReplyTo>.

Si el solicitante compila MAP incorrectas y que provocan fallos de validación, CICS envía el mensaje de error de vuelta al solicitante en lugar de a la dirección especificada por la MAP <wsa:FaultTo>.

Solicitud y respuesta síncronas

De forma predeterminada, la parte de respuesta de un mensaje bidireccional se devuelve según el protocolo subyacente en uso. En el caso de una solicitud HTTP, la respuesta se devuelve síncronamente en la respuesta HTTP.

Solicitud y respuesta asíncronas

Una respuesta asíncrona se dirige a otro servicio web y no llega de vuelta a la aplicación de solicitante original. En el caso de una solicitud HTTP, la conexión con el cliente solicitante se cierra con una respuesta HTTP 202. Si el proveedor de servicios web se está ejecutando en un sistema CICS, la aplicación de solicitante recibirá un mensaje de respuesta vacío. Si el proveedor de servicios web se está ejecutando en un sistema WebSphere MQ, la aplicación de solicitante no recibirá ninguna respuesta.

Para cambiar el destino de la parte de respuesta de un mensaje bidireccional, debe especificar las direcciones adecuadas en la MAP <wsa:ReplyTo> o las MAP <wsa:ReplyTo> y <wsa:FaultTo>.

Para obtener una lista completa de las MAP que son obligatorias en WSDL 1.1 y WSDL 2.0, consulte “Propiedades de direccionamiento de mensajes obligatorias para WS-Addressing”.

Propiedades de direccionamiento de mensajes obligatorias para WS-Addressing

La especificación de metadatos de WS-Addressing 1.0 indica que las propiedades de direccionamiento de mensajes (MAPs) deben incluirse en documentos WSDL 1.1 y WSDL 2.0. La implementación de CICS de WS-Addressing le ayuda a cumplir con las especificaciones de WS-Addressing proporcionando automáticamente valores para estas MAP obligatorias.

Puede especificar sus propios valores para las MAP en el WSDL que proporciona, y puede actualizar estos valores en el contexto de direccionamiento utilizando los mandatos de API de CICS WS-Addressing. Si no proporciona valores para las MAP obligatorias, CICS generará los valores.

La tabla siguiente listas las MAP que son obligatorias para los diferentes patrones de intercambio de mensajes soportados (MEPs) con WSDL 1.1 y WSDL 2.0:

Tabla 14. Propiedades de direccionamiento de mensajes obligatorias para WS-Addressing

Nombre de MAP de WS-Addressing	Descripción	Obligatoria en WSDL 1.1	Obligatoria en WSDL 2.0
<wsa:To>	La dirección del destinatario deseado del mensaje.	No	No

Tabla 14. Propiedades de direccionamiento de mensajes obligatorias para WS-Addressing (continuación)

Nombre de MAP de WS-Addressing	Descripción	Obligatoria en WSDL 1.1	Obligatoria en WSDL 2.0
<wsa:Action>	La acción de WS-Addressing: entrada, salida o error.	Obligatoria para los siguientes MEP: Unidireccional De dos direcciones (Solicitud) De dos direcciones (Respuesta)	Obligatoria para los siguientes MEP: Sólo entrada Sólo entrada sólida (Entrada) Sólo entrada sólida (Error) Entrada-salida (Entrada) Entrada-salida (Salida) Entrada-opcional-salida (Entrada) Entrada-opcional-salida (Salida)
<wsa:From>	El punto final desde el que se origina el mensaje.	No	No
Este	valor	no	es necesario
<wsa:ReplyTo>	El punto final de un destinatario deseado para respuestas al mensaje.	No	No
<wsa:FaultTo>	El punto final del destinatario deseado para errores relacionados con el mensaje.	No	No
<wsa:MessageID>	Un identificador de mensajes exclusivo.	Obligatoria para los siguientes MEP: De dos direcciones (Solicitud)	Obligatoria para los siguientes MEP: Sólo entrada sólida (Entrada) Entrada-salida (Entrada) Entrada-opcional-salida (Entrada)
<wsa:RelatesTo>	Un par de valores que indica cómo se relaciona este mensaje con otro mensaje. Este elemento incluye el <wsa:MessageID> del mensaje relacionado y un atributo opcional transmite el tipo de relación.	Obligatoria para los siguientes MEP: De dos direcciones (Respuesta)	Obligatoria para los siguientes MEP: Sólo entrada sólida (Error) Entrada-salida (Salida) Entrada-opcional-salida (Salida)

Para obtener más información, consulte la especificación *W3C WS-Addressing 1.0 Metadata*: <http://www.w3.org/TR/ws-addr-metadata>.

Notas:

- Si no se establece un valor para el elemento de dirección de la MAP <wsa:ReplyTo>, la dirección se establece en el URI anónimo: <http://www.w3.org/2005/08/addressing/anonymous>. El URI anónimo indica que las respuestas se envían de vuelta al solicitante.
- Si no se especifica un valor para el elemento de dirección de la MAP <wsa:FaultTo>, CICS establece esta dirección en el mismo valor que el elemento de dirección de la MAP <wsa:ReplyTo>.

Tenga en cuenta que si el solicitante compila MAP incorrectas y que provocan fallos de validación, CICS envía el mensaje de error de vuelta al solicitante en lugar de a la dirección especificada por la MAP <wsa:FaultTo>.

- Si no se especifica el valor de la MAP <wsa:To> MAP, CICS establece la dirección en el URI anónimo: <http://www.w3.org/2005/08/addressing/anonymous>. El URI anónimo indica que la solicitud se enviará a la dirección especificada en el contenedor DFHWS-URI; para obtener más información, consulte “Contenedor DFHWS-URI” en la página 166.
- Puede definir las MAP <wsa:Action> específicamente en el documento WSDL, o puede permitir que CICS las genere automáticamente.
- CICS establece automáticamente un valor exclusivo para la MAP <wsa:MessageID> en tiempo de ejecución para mensajes de solicitud que esperan una respuesta y para mensajes de respuesta.
- La MAP <wsa:RelatesTo> es obligatoria para los mensajes de respuesta. El tipo de relación del mensaje es opcional y va siempre a <http://www.w3.org/2005/08/addressing/reply>.

Seguridad de Web Services Addressing

Las comunicaciones que viajan por una red pública utilizando Web Services Addressing (WS-Addressing) deben estar aseguradas adecuadamente y se debe establecer un nivel de confianza suficiente entre las partes en comunicación. Se le recomienda que utilice seguridad a nivel de transporte, como SSL o HTTPS, para proteger sus comunicaciones.

La seguridad a nivel de transporte, como SSL o HTTPS, es la forma más sencilla de garantizar que sus comunicaciones WS-Addressing sean seguras. Si la seguridad a nivel de transporte no está disponible, puede proteger sus mensajes firmando las propiedades de direccionamiento de mensajes WS-Addressing y cifrando las referencias de punto final.

CICS no puede firmar cabeceras que contienen propiedades de direccionamiento de mensajes WS-Addressing o cifrar referencias de punto final. Sin embargo, CICS puede verificar firmas en mensajes entrantes y puede descifrar cabeceras que se han cifrado. Si desea utilizar la firma y cifrado para proteger sus comunicaciones, debe utilizar una pasarela de seguridad externa, como IBM WebSphere DataPower XML Security Gateway. Para obtener más información, consulte el apartado IBM WebSphere DataPower XML Security Gateway.

Ejemplo de Web Services Addressing

Este ejemplo proporciona una descripción general de alto nivel del proceso que tiene lugar cuando un cliente hace un pedido en una empresa que utiliza Web Services Addressing para enviar mensajes.

Una compañía internacional que vende componentes electrónicos utiliza Web Services Addressing en su negocio. La infraestructura de esta compañía consiste en un Cliente que hace un pedido, un Grupo de servicios de distribución, un Servicio de cumplimiento y un Servicio de configuración.

Utilizando WS-Addressing ofrece a la compañía los siguientes beneficios:

- WS-Addressing proporciona un mecanismo de transporte independiente para transferir mensajes, esto fomenta la interoperabilidad entre servicios web que se ejecutan en diferentes plataformas. En este ejemplo, los servicios de distribución propiedad de la compañía se ejecutan en diversas plataformas; WS-Addressing hace sencilla la interoperabilidad entre diferentes plataformas porque los

proveedores y solicitantes de servicio web no necesitan ser conscientes de la plataforma en la que se está ejecutando el servicio con el que están intercambiando mensajes.

- WS-Addressing se puede utilizar para cambiar el destino del mensaje de respuesta actualizando la EPR en la MAP <wsa:ReplyTo>. En este ejemplo, el Servicio de cumplimiento modifica el destino del mensaje de respuesta cuando selecciona el Servicio de distribución al que se desvía el mensaje.

La compañía tiene varios centros de distribución en diferentes países; cada uno de los centros de distribución está representado en este ejemplo por un Servicio de distribución y se registra con el Servicio de configuración.

El Servicio de cumplimiento selecciona cuál es el Servicio de distribución más adecuado para procesar el pedido basándose en varios factores, que pueden incluir la disponibilidad de los artículos solicitados y la distancia del Centro de distribución al cliente.

La información de direccionamiento se pasa a y desde el Servicio de configuración. El Servicio de configuración almacena las direcciones de los servicios disponibles en la forma de Referencias de punto final. Los nuevos servicios se registran con el Servicio de configuración creando una EPR utilizando el mandato **WSAEPR CREATE** y enviando la EPR al Servicio de configuración. El Servicio de configuración requiere la EPR como un bloque de XML, por lo tanto el parámetro **WSADDR-EPR-ANY** en DFHWS2LS debe establecerse en TRUE. La opción **WSADDR-EPR-ANY=TRUE** se va a utilizar para indicar a CICS que trate la EPR como un elemento <xsd:any>; CICS debe colocarla en un contenedor en lugar de transformarla en una estructura de lenguaje en tiempo de ejecución.

La forma en la que estos servicios interactúan se muestra en el siguiente diagrama. El diagrama muestra otros servicios que se han excluido de la tarea, y que pueden ser relevantes en la aplicación empresarial:

- Un Servicio de seguimiento, que se puede actualizar por medio de los demás servicios con el estado del pedido.
- Un Servicio de resolución de problemas para manejar los mensajes de error que surgen.
- Un servicio de devolución de llamadas al cliente de pedidos para manejar los mensajes de respuesta dirigidos al cliente de pedidos.

- a. El Servicio de cumplimiento utiliza el mandato **WSACONTEXT GET** para extraer la referencia del pedido y otras propiedades de direccionamiento del contexto de direccionamiento.
- b. El Servicio de cumplimiento selecciona el Servicio de distribución más adecuado del Servicio de configuración.
- c. Se añade la EPR <wsa:ReplyTo> al contexto de direccionamiento:

```
<wsa:EndpointReference
  xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <wsa:Address>http://www.example.ibm.com/DistributionService</wsa:Address>
</wsa:EndpointReference>
```

El Servicio de direccionamiento utiliza el mandato **WSACONTEXT BUILD** para añadir la EPR de ReplyTo del Servicio de distribución elegido al contexto de direccionamiento de la solicitud.

- d. El Servicio de cumplimiento utiliza el mandato **WSACONTEXT BUILD** repetidamente para añadir la referencia de pedido y otra información al contexto de direccionamiento de la solicitud.
 - e. Se añade un contenedor DFHNORESPONSE a la interconexión del Cliente de pedidos para indicarle que no recibirá una respuesta y que el mensaje de respuesta se redirige en forma de mensaje de solicitud al Servicio de distribución.
4. El Servicio de distribución recibe el mensaje de respuesta redirigido y procesa el pedido.
 - a. El Servicio de distribución utiliza un mandato **WSACONTEXT GET** para extraer los detalles de direccionamiento y referencia del pedido del contexto de direccionamiento de la solicitud.
 - b. El Servicio de distribución procesa el pedido.

Ejemplo de <wsa:To>

1. El Cliente de pedidos solicita la EPR del servicio al que desea enviar un mensaje desde el Servicio de configuración. En este ejemplo, el Cliente de pedidos solicita la EPR del Servicio de cumplimiento.
2. El Servicio de configuración crea y envía un mensaje de respuesta:
 - a. El Servicio de configuración crea la EPR <wsa:To> solicitada para el Servicio de cumplimiento utilizando el mandato de API **WSAEPR CREATE: EXEC CICS WSAEPR CREATE**.
 - b. El Servicio de configuración escribe la salida desde el mandato **WSAEPR CREATE** a un contenedor: EXEC CICS PUT CONTAINER(work-cont).
 - c. El Servicio de configuración copia el nombre de contenedor en el elemento myEpr-xml-cont: MOVE work-cont TO myEpr-xml-cont.
 - d. El Servicio de configuración envía un mensaje de respuesta al Cliente de pedidos, este mensaje contiene el contenido del contenedor nombrado por el contenedor myEpr-xml-cont. En este ejemplo, el contenido del contenedor work-cont se envía al cliente de pedidos dentro del elemento <wsa:myEpr>:

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  ...
  <env:Body>
    <wsa:myEpr>
      <wsa:EndpointReference>
        <wsa:Address>
          Fulfilment_Service_EPR_XML
        </wsa:Address>
      </wsa:EndpointReference>
```

```

        </wsa:myEpr>
    </env:Body>
    ...
</env:Envelope>

```

Figura 27 muestra el intercambio de mensajes de solicitud/respuesta entre el Cliente de pedidos y el Servicio de configuración. Este intercambio de mensajes implica dos interconexiones de servicios web típicas.

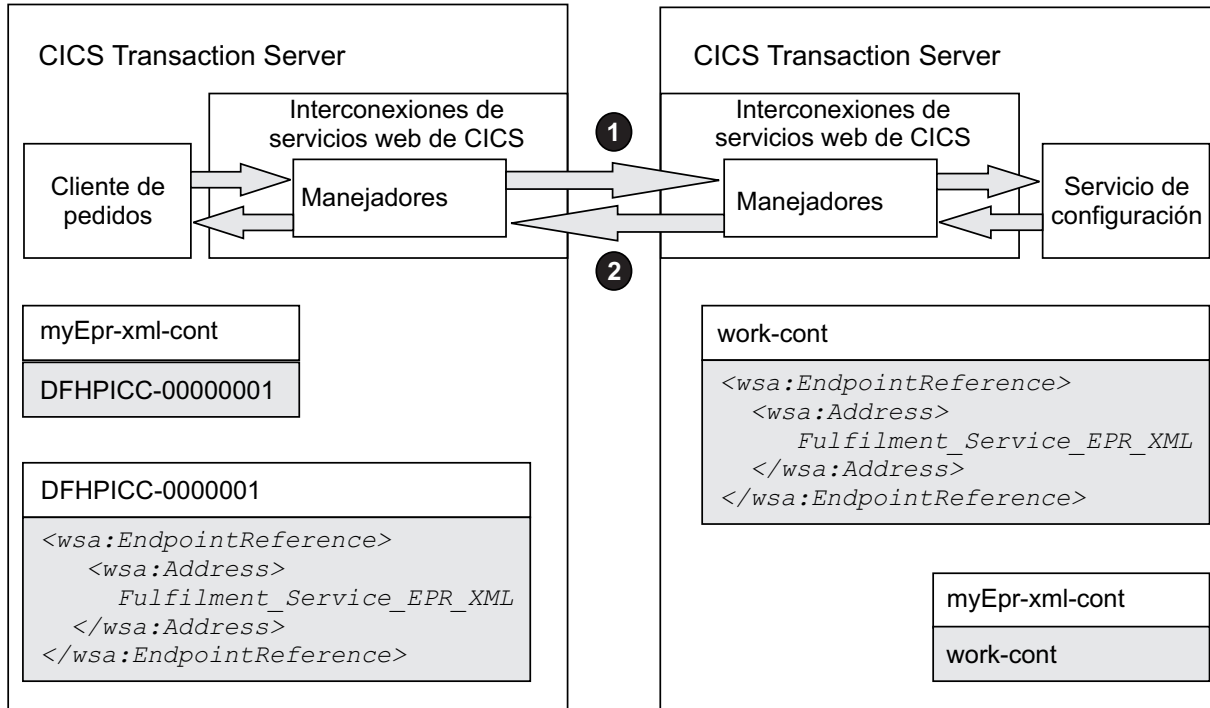


Figura 27. Intercambio de mensajes de solicitud-respuesta entre el Cliente de pedidos y el Servicio de configuración

3. El Cliente de pedidos recibe el mensaje de respuesta, compila la EPR <wsa:To> y envía una solicitud al Servicio de cumplimiento:
 - a. El Cliente de pedidos extrae los datos de EPR <wsa:To> del mensaje de respuesta.
 - b. CICS llena un contenedor exclusivo, en este ejemplo el contenedor DFHPICC-00000001, con los datos de EPR <wsa:To>.
 - c. CICS copia el nombre del contenedor, en este ejemplo DFHPICC-00000001, en el elemento myEpr-xml-cont.
 - d. El Cliente de pedidos lee el contenido del contenedor especificado por el elemento myEpr-xml-cont y lo proporciona como entrada para el mandato de API **WSACONTEXT BUILD**. El mandato **WSACONTEXT BUILD** utiliza la entrada para compilar la EPR <wsa:To> para el Servicio de cumplimiento.
 - e. El Cliente de pedidos emite un mandato **INVOKE SERVICE** que inicia el proceso de interconexión.
 - f. El manejador de direccionamiento de servicios web CICS, DFHWSADH, en la interconexión de salida convierte la EPR <wsa:To> en una dirección y un

conjunto opcional de parámetros de referencia que coloca en la cabecera del mensaje de solicitud SOAP que se está enviando al Servicio de cumplimiento:

```
<env:Header>
  <wsa:To>http://example.ibm.com/Fulfilment_Service</wsa:To>
</env:Header>
```

Figura 28 muestra la solicitud desde el Cliente de pedidos al Servicio de cumplimiento. Esta solicitud implica una interconexión de servicios web que incluye el manejador de direccionamiento de servicios web CICS, DFHWSADH.

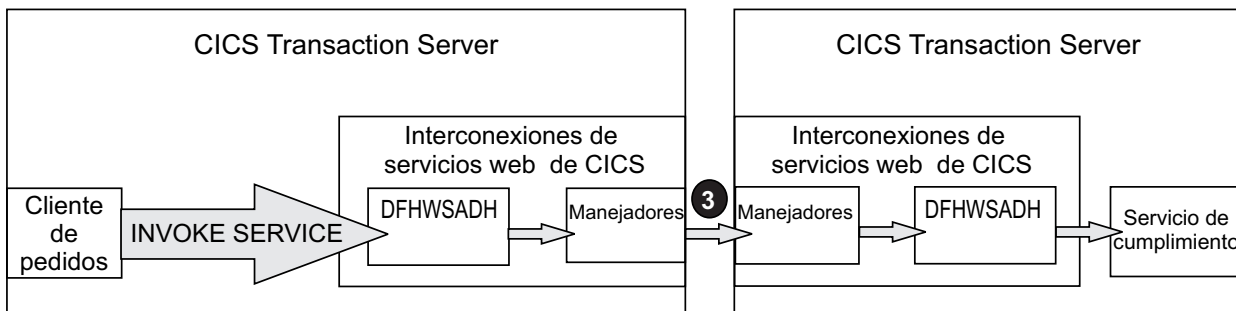


Figura 28. Solicitud desde el Cliente de pedidos al Servicio de cumplimiento

Terminología de Web Services Addressing

Términos utilizados para explicar el soporte para el direccionamiento de los servicios Web Services (WS-Addressing).

contexto de direccionamiento

Un documento XML que almacena propiedades de direccionamiento de mensajes WS-Addressing (MAPs) antes de que se envíen en mensajes de solicitud SOAP y después de que se reciban de solicitudes SOAP y mensajes de respuesta.

referencia de punto final (EPR)

Una estructura XML que contiene información de direccionamiento que se utiliza para dirigir un mensaje a un servicio web. Esta información de direccionamiento incluye la dirección de destino del mensaje, parámetros de referencia opcionales para que los utilice la aplicación y metadatos opcionales.

propiedad de direccionamiento de mensaje (MAP)

Un elemento XML que transmite información de direccionamiento para un mensaje de servicio web específico, como un ID de mensaje exclusivo, el destino del mensaje y referencias de punto final del mensaje.

Soporte para SAML

CICS soporta el uso de Security Assertion Markup Language (SAML) para describir e intercambiar información de seguridad entre socios comerciales en línea.

CICS soporta los estándares SAMLCore1.1 y SAML Core2.0. No soporta los protocolos que se describen en esos estándares.

Puede configurar las interconexiones de solicitante y proveedor para utilizar señales SAML, pero primero debe desplegar el CICS Security Token Service (STS). Para obtener más información sobre la configuración de la instalación de CICS para soportar SAML, consulte Configuración de CICS para SAML.

Capítulo 3. Desarrollo de servicios web

Creación de un servicio web JSON

Puede exponer las aplicaciones CICS existentes como servicios web JSON y crear nuevas aplicaciones CICS para que actúen como proveedores de servicios web.

Antes de empezar

Antes de empezar a crear un servicio web JSON, debe configurar el sistema CICS para dar soporte a servicios web JSON. Para obtener más información, consulte el apartado “Creación de la infraestructura de CICS para un proveedor de servicios JSON” en la página 69.

Acerca de esta tarea

El asistente de CICS JSON es un programa de utilidad proporcionado que le ayuda a crear los artefactos necesarios para una nueva aplicación de proveedor de servicios web JSON nueva, o para habilitar una aplicación existente como proveedor de servicios web JSON.

El asistente de CICS JSON puede crear un esquema JSON a partir de una estructura de lenguaje de alto nivel o una estructura de lenguaje de alto nivel desde un esquema JSON existente; soporta COBOL, C/C++ y PL/I. También genera información que se utiliza para habilitar la conversión en tiempo de ejecución automática de mensajes JSON a contenedores y COMMAREAs y viceversa. El soporte de servicios web de CICS JSON utiliza esta información durante el proceso de interconexión.

Cree su servicio web JSON, como se describe en el siguiente procedimiento y valide que funciona correctamente:

Procedimiento

1. Cree un servicio web JSON. Utilice el asistente de JSON para crear las estructuras de lenguaje o esquema JSON y desplegarlas en CICS. Utilice el mandato **PIPELINE SCAN** para crear automáticamente los recursos CICS requeridos.
2. Inicie el servicio web JSON para probar que funciona como esperaba.

Qué hacer a continuación

Estos pasos se explican de forma más detallada en los temas siguientes.

El asistente de JSON de CICS

El asistente de JSON de CICS es un conjunto de programas de utilidad por lotes que crea una correlación entre estructuras de lenguaje y esquemas JSON. CICS utiliza esta correlación durante el tiempo de ejecución para hacer la transformación entre datos de aplicación y JSON. El asistente soporta un despliegue rápido de aplicaciones CICS para utilizarlas en proveedores de servicios y solicitantes de servicio, con un mínimo esfuerzo de programación.

Cuando utiliza el asistente de JSON para CICS, no tiene que escribir su propio código para analizar mensajes de entrada y para construir mensajes de salida; CICS correlaciona datos entre el mensaje de JSON y la estructura de datos del programa de aplicación.

El asistente puede crear un esquema JSON desde una estructura de lenguaje de alto nivel o una estructura de lenguaje de alto nivel desde el esquema JSON existente, y soporta COBOL, C/C++ y PL/I. También genera información utilizada para habilitar la conversión en tiempo de ejecución automática de los mensajes JSON para contenedores y COMMAREAs, y viceversa.

El asistente de JSON de CICS consta de dos programas de utilidad:

DFHLS2JS

Genera un archivo de enlace de servicio web desde una estructura de lenguaje. Este programa de utilidad también genera un esquema JSON.

DFHJS2LS

Genera un archivo de enlace de servicio web desde un esquema JSON. Este programa de utilidad también genera una estructura de lenguaje que puede utilizar en los programas de aplicación.

Los procedimientos JCL para ejecutar ambos programas están en la biblioteca *hlq*.XDFHINST .

La modalidad de uso relevante para el procedimiento DFHLS2JS o DFHJS2LS depende de los requisitos:

- DFHLS2JS: conversión de lenguaje de alto nivel a esquema JSON para interfaz enlazable
- DFHJS2LS: conversión de esquema JSON a lenguaje de alto nivel para interfaz enlazable
- DFHLS2JS: conversión de lenguaje de alto nivel a esquema JSON para servicios de solicitud-respuesta
- DFHJS2LS: conversión de esquema JSON a lenguaje de alto nivel para servicios solicitud-respuesta
- DFHJS2LS: Conversión de esquema JSON a un lenguaje de alto nivel para servicios RESTful

Para obtener más información sobre los programas de utilidad del asistente de JSON y las correlaciones de datos, consulte los temas siguientes.

DFHLS2JS: Conversión de lenguaje de alto nivel a esquema JSON para servicios de solicitud-respuesta

El procedimiento de DFHLS2JS genera un archivo de esquema JSON a partir de la estructura de datos de lenguaje de alto nivel. Puede utilizar DFHLS2JS cuando expone un programa de aplicación CICS como proveedor de servicios.

Las sentencias de control de trabajos para DFHLS2JS, sus parámetros simbólicos, sus parámetros de entrada y sus descripciones y un trabajo de ejemplo le ayudan a utilizar este procedimiento.

El procedimiento DFHLS2JS JCL está instalado en el conjunto de datos *HLQ*.XDFHINST , donde *HLQ* es el calificador de alto nivel donde está instalado CICS.

Sentencias de control de trabajos para DFHLS2JS

JOB Inicia el trabajo.

EXEC Especifica el nombre del procedimiento (DFHLS2JS).

INPUT.SYSUT1 DD

Especifica la entrada. Los parámetros de entrada se especifican normalmente en el flujo de entrada. Sin embargo, se pueden definir en un conjunto de datos en miembro de un conjunto de datos particionados.

Parámetros simbólicos

Los siguientes parámetros simbólicos se definen en DFHLS2JS:

JAVADIR = *vía de acceso*

Especifica el nombre del directorio Java que utiliza DFHLS2JS. El valor de este parámetro se añade a /usr/lpp/ para crear un nombre de vía de acceso completo de /usr/lpp/ *vía de acceso* .

Generalmente, no es necesario especificar este parámetro. El valor predeterminado es el valor proporcionado para el trabajo de instalación (DFHISTAR) de CICS en el parámetro **JAVADIR**.

PATHPREF = *prefijo*

Especifica un prefijo que amplía la vía de acceso del directorio z/OS UNIX que se utiliza en otros parámetros o '' (serie vacía) si no se utiliza un prefijo.

Generalmente, no es necesario especificar este parámetro. El valor predeterminado es el valor proporcionado para el trabajo de instalación (DFHISTAR) de CICS en el parámetro **PATHPREF**.

SERVICE = *valor*

Utilice este parámetro sólo cuando el soporte de IBM le indique que lo haga.

TMPDIR = *tmpdir*

Especifica la ubicación de un directorio en z/OS UNIX que DFHLS2JS utiliza como espacio de trabajo temporal. El ID de usuario utilizado para ejecutar el trabajo debe tener permisos de lectura y escritura para este directorio.

El valor predeterminado es /tmp.

TMPFILE = *tmpprefix*

Especifica un prefijo que DFHLS2JS utiliza para construir los nombres de los archivos de espacio de trabajo temporal.

El valor predeterminado es LS2JS.

USSDIR = *vía de acceso*

Especifica el nombre del directorio CICS TS en el sistema de archivos de los servicios del sistema UNIX. El valor de este parámetro se añade a /usr/lpp/cicsts/ para crear un nombre de vía de acceso completo de /usr/lpp/cicsts/ *vía de acceso* . Esto debe especificarse como '.' (punto) si se utiliza el valor predeterminado.

Generalmente, no es necesario especificar este parámetro. El valor predeterminado es el valor proporcionado para el trabajo de instalación (DFHISTAR) de CICS en el parámetro **USSDIR**.

El espacio de trabajo temporal

DFHLS2JS crea estos tres archivos temporales en tiempo de ejecución:

tmpdir / *tmpprefix* .in

```
tmpdir / tmpprefix .out
tmpdir / tmpprefix .err
```

donde:

tmpdir es el valor que se especifica en el parámetro **TMPDIR**.

tmpprefix es el valor que se especifica en el parámetro **TMPFILE**.

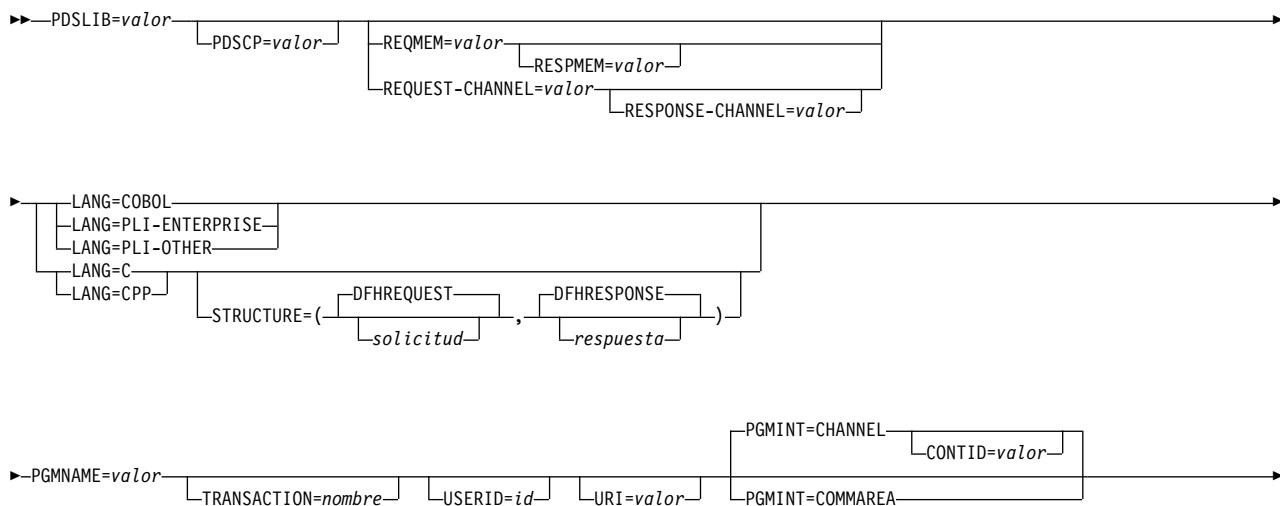
Los nombres predeterminados para los archivos, cuando no se especifica **TMPDIR** y **TMPFILE** son los siguientes:

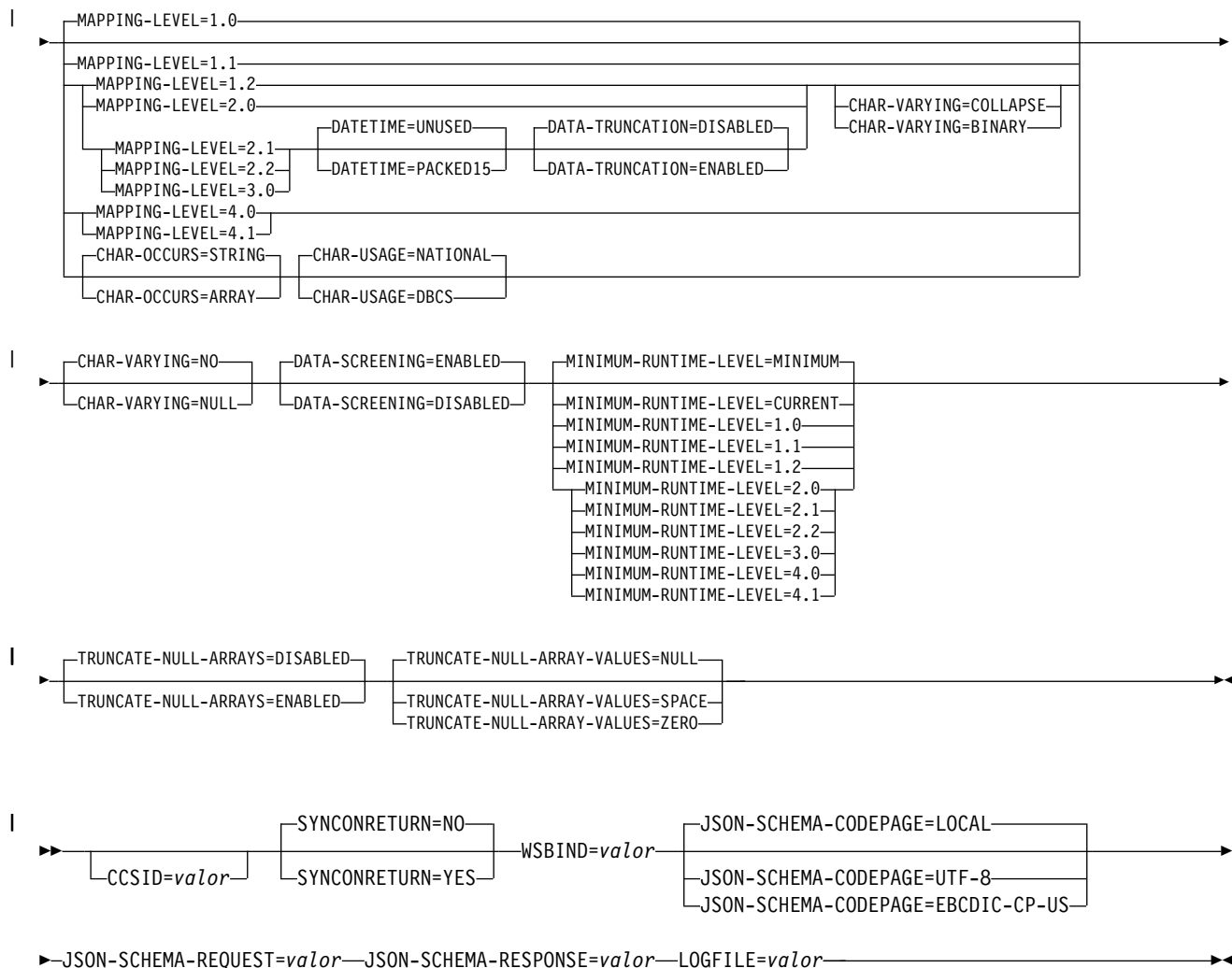
```
/tmp/LS2JS.in
/tmp/LS2JS.out
/tmp/LS2JS.err
```

Importante: DFHLS2JS no bloquea el acceso a los archivos z/OS UNIX o los miembros del conjunto de datos. Por lo tanto, si dos o más instancias de DFHLS2JS se ejecutan de forma simultánea y utilizan los mismos archivos de espacio de trabajo temporal, nada impide a un trabajo sobrescribir los archivos de espacio de trabajo mientras otro trabajo los está utilizando, llevando a errores imprevisibles.

Por lo tanto, se aconseja crear un convenio de denominación y procedimientos operativos que eviten esta situación. Por ejemplo, puede utilizar el parámetro simbólico de sistema **SYSUID** para generar nombres de archivo de espacios de trabajo que son exclusivos para un usuario individual. Estos archivos temporales se suprimen antes de la finalización del trabajo.

Parámetros de entrada de DFHLS2JS





Utilización de parámetros

- Puede especificar los parámetros de entrada en cualquier orden.
- Cada parámetro se debe iniciar en una nueva línea.
- Un parámetro, y su carácter de continuación, si utiliza uno, no deben sobrepasar la columna 72; las columnas 73 a 80 deben contener espacios en blanco.
- Si un parámetro es demasiado largo para ajustarse en una única línea, utilice el carácter de asterisco (*) al final de la línea para indicar que el parámetro continúa en la línea siguiente. Todo, incluidos los espacios antes del asterisco, se considera parte del parámetro. Por ejemplo:

```
WSBIND=wsbinddir*
/app1
```

es equivalente a

```
WSBIND=wsbinddir/app1
```

- Un carácter # en la primera posición de carácter de la línea es un carácter de comentario. La línea se ignora.
- Una coma en la última posición de carácter de la línea es un separador de línea opcional y se ignora.

Descripciones de parámetros

CCSID = *valor*

Especifica el CCSID que se utiliza en tiempo de ejecución para codificar datos de caracteres en la estructura de datos de la aplicación. El valor de este parámetro sustituye el valor del parámetro de inicialización del sistema **LOCALCCSID**. El *valor* debe ser un EBCDIC CCSID soportado por Java y . Si no especifica este parámetro, la estructura de datos de aplicación se codifica utilizando el CCSID especificado en el parámetro de inicialización del sistema.

CHAR-VARYING = { **NO** | **NULL** | **COLLAPSE** | **BINARY** }

Especifica cómo los campos de carácter de la estructura de lenguaje se correlacionan cuando el nivel de correlación es 1.2 o superior. Un campo de carácter en COBOL es una cláusula Picture de tipo X, por ejemplo PIC(X) 10 ; un campo de carácter en C/C++ es una matriz de caracteres. Puede seleccionar estas opciones:

NO Los campos de carácter se correlacionan con una serie JSON y se procesan como campos de longitud fija. La longitud máxima de los datos es igual a la longitud del campo. NO es el valor predeterminado para el parámetro **CHAR-VARYING** para COBOL y PL/I a niveles de correlación 2.0 y anteriores.

Este valor no se aplica a estructuras de lenguaje Enterprise y Otros PL/I.

NULL Los campos de carácter se correlacionan con una serie JSON y se procesan como series terminadas en nulo. CICSañade un carácter nulo de terminación cuando se está transformando desde un mensaje JSON. La longitud máxima de la serie de caracteres se calcula como un carácter menos que la longitud indicada en la estructura de lenguaje. NULL es el valor predeterminado para el parámetro **CHAR-VARYING** para C/C++.

Este valor no se aplica a estructuras de lenguaje Enterprise y Otros PL/I.

COLLAPSE

Los campos de caracteres se correlacionan con una serie JSON. Los espacios en blanco finales del campo no se incluyen en el mensaje JSON. El mensaje JSON de entrada se analiza para eliminar todos los espacios en blanco iniciales, finales e incluidos. COLLAPSE es el valor predeterminado para el parámetro **CHAR-VARYING** para COBOL y PL/I a nivel de correlación 2.1 y en adelante.

BINARY

Los campos de carácter se correlacionan con una serie JSON que contiene datos de codificados en base64 y se procesan como campos de longitud fija. El valor BINARY en el parámetro **CHAR-VARYING** está disponible sólo en niveles de correlación 2.1 y en adelante.

CHAR-OCCURS = { **STRING** | **ARRAY** }

Especifica cómo las matrices de caracteres en la estructura de lenguaje se correlacionan cuando el nivel de correlación es 4.0 o superior. Por ejemplo, PIC X OCCURS 20. Este parámetro sólo lo utiliza el lenguaje COBOL.

ARRAY

Las matrices de caracteres se correlacionan con la matriz JSON. Esto significa que cada carácter se correlaciona como un elemento JSON individual. Este también es el comportamiento a niveles de correlación 3.0 y anteriores.

STRING

Las matrices de caracteres se correlacionan con una serie JSON. Esto significa que toda la matriz COBOL se correlaciona como un único elemento JSON.

CHAR-USAGE = { NATIONAL | DBCS }

En COBOL, el tipo de datos nacional, PIC N , se puede utilizar para datos UTF-16 o DBCS. Este valor lo controla la opción del compilador NSYMBOL. Debe establecer el parámetro **CHAR-USAGE** en el asistente en el mismo valor que la opción de compilador NSYMBOL para asegurarse de que los datos se manejan adecuadamente. Se establece normalmente en CHAR-USAGE=NATIONAL cuando utiliza UTF-16.

DBCS Los datos de los campos PIC (*n*) se tratan como datos cifrados en DBCS.

NATIONAL

Los datos de los campos PIC (*n*) se tratan como datos cifrados en UTF-16.

CONTID = *valor*

En un proveedor de servicios, especifique el nombre del contenedor que tiene la estructura de datos de nivel superior utilizada para representar un mensaje JSON.

La longitud del contenedor que CICS pasa al programa de aplicación de destino es mayor que las longitudes del contenedor de solicitud y el contenedor de respuesta.

DATA-SCREENING = { ENABLED | DISABLED }

Especifica si los datos proporcionados por la aplicación se analizan para detectar errores.

ENABLED

Los datos de tiempo de ejecución proporcionados por la aplicación que son incoherentes con la estructura de lenguaje, se tratan como un error y se emite un mensaje DFHPI1010. Se devuelve una respuesta de error a la aplicación.

DISABLED

Los valores de datos de los datos de tiempo de ejecución proporcionados por la aplicación que son incoherentes con la estructura de lenguaje se sustituyen por valores predeterminados. Por ejemplo, un cero sustituye un valor erróneo en un campo numérico. No se emite mensaje DFHPI1010 y se devuelve una respuesta normal a la aplicación. Esta característica se puede utilizar para evitar las respuestas de error INVALID_PACKED_DEC y INVALID_ZONED_DEC generadas desde los campos de salida no inicializados.

DATA-TRUNCATION = { DISABLED | ENABLED }

Especifica si los datos de longitud variable se toleran en una estructura de campo de longitud fija:

DISABLED

Si los datos son menores que la longitud fija que está esperando CICS, CICS rechaza los datos truncados y emite un mensaje de error.

ENABLED

Si los datos son menores que la longitud fija que está esperando CICS, CICS tolera los datos truncados y procesa los datos que faltan como valores nulos

DATETIME = { **UNUSED** | **PACKED15** }

Especifica si los campos ABSTIME potenciales en la estructura de lenguaje de alto nivel se correlacionan como indicaciones de fecha y hora:

PACKED15

Los campos decimales empaquetados de longitud 15 (8 bytes) se tratan como campos CICS ABSTIME y se correlacionan como indicaciones de fecha y hora.

UNUSED

Los campos decimales empaquetados de longitud 15 (8 bytes) no se tratan como indicaciones de fecha y hora.

Este parámetro se puede definir en un nivel de correlación de 3.0.

JSON-SCHEMA-CODEPAGE = { **LOCAL** | **UTF-8** | **EBCDIC-CP-US** }

Especifica la página de códigos que se utiliza para generar los documentos de esquema JSON.

LOCAL

Especifica que los esquemas JSON se generan utilizando la página de códigos predeterminada para el sistema de archivos.

UTF-8 Especifica que los esquemas JSON se generan utilizando la página de códigos UTF-8.

EBCDIC-CP-US

Especifica que los esquemas JSON se generan utilizando la página de códigos US EBCDIC.

JSON-SCHEMA-REQUEST = *valor*

Este es un parámetro obligatorio.

El valor indica la ubicación de los servicios del sistema de UNIX donde se almacena el esquema JSON de solicitud.

JSON-SCHEMA-RESPONSE = *valor*

Este es un parámetro obligatorio.

El valor indica la ubicación de los servicios del sistema de UNIX donde se almacena el esquema JSON de respuesta.

LANG = **COBOL**

Especifica que el lenguaje de programación de la estructura de lenguaje de alto nivel es COBOL.

LANG = **PLI-ENTERPRISE**

Especifica que el lenguaje de programación de la estructura de lenguaje de alto nivel es Enterprise PL/I.

LANG = **PLI-OTHER**

Especifica que el lenguaje de programación de la estructura de lenguaje de alto nivel es un nivel de PL/I distinto a Enterprise PL/I.

LANG = **C**

Especifica que el lenguaje de programación de la estructura de lenguaje de alto nivel es C.

LANG = CPP

Especifica que el lenguaje de programación de la estructura de lenguaje de alto nivel es C++.

LOGFILE = valor

El nombre completo del archivo de z/OS UNIX en el que DFHLS2JS escribe la información de rastreo y registro de actividad. DFHLS2JS crea el archivo pero no la estructura de directorio, si no existe.

Normalmente, no utiliza este archivo, pero puede solicitarlo la organización de servicio de IBM si encuentra problemas con DFHLS2JS.

MAPPING-LEVEL = { 1.0 | 1.1 | 1.2 | 2.0 | 2.1 | 2.2 | 3.0 | 4.0 | 4.1 }

Especifica el nivel de correlación que utiliza DFHLS2JS cuando genera el archivo de enlace de servicio y esquema JSON. Puede seleccionar estas opciones:

- 1.0** Esta opción se retiene para la compatibilidad con servicios web SOAP. No se recomienda utilizar con JSON.
- 1.1** Esta opción se retiene para la compatibilidad con servicios web SOAP. No se recomienda utilizar con JSON.
- 1.2** Esta opción se retiene para la compatibilidad con servicios web SOAP. No se recomienda utilizar con JSON.
- 2.0** Esta opción se retiene para la compatibilidad con servicios web SOAP. No se recomienda utilizar con JSON.
- 2.1** Esta opción se retiene para la compatibilidad con servicios web SOAP. No se recomienda utilizar con JSON.
- 2.2** Esta opción se retiene para la compatibilidad con servicios web SOAP. No se recomienda utilizar con JSON.
- 3.0** Utilice este nivel de correlación para generar un esquema JSON utilizando el conjunto de opciones completo disponible.
- 4.0** Utilice este nivel de correlación con una región de CICS TS 5.2 o región posterior. En este nivel de correlación, puede utilizar campos de COBOL OCCURS DEPENDING ON y el parámetro **CHAR-OCCURS**.
- 4.1** Utilice este nivel de correlación para el soporte de matriz truncable, con una región de CICS TS V5.2 que tiene aplicado APAR PI67641, o una región de CICS TS V5.3 o posterior.

Para obtener más información sobre los niveles de correlación, consulte Niveles de correlación para los asistentes de JSON de CICS .

MINIMUM-RUNTIME-LEVEL = { MINIMUM | 1.0 | 1.1 | 1.2 | 2.0 | 2.1 | 2.2 | 3.0 | 4.0 | 4.1 | CURRENT }

Especifica el entorno de ejecución de CICS mínimo en el que se puede desplegar el archivo de enlace de servicio web. Si selecciona un nivel que no coincide con los demás parámetros que ha especificado, recibe un mensaje de error. Puede seleccionar estas opciones:

MINIMUM

El nivel de tiempo de ejecución más bajo posible de CICS se asigna automáticamente a los parámetros determinados que seleccione.

- 1.0** Esta opción se retiene para la compatibilidad con servicios web SOAP. No se recomienda utilizar con JSON.

- 1.1 Esta opción se retiene para la compatibilidad con servicios web SOAP. No se recomienda utilizar con JSON.
- 1.2 Esta opción se retiene para la compatibilidad con servicios web SOAP. No se recomienda utilizar con JSON.
- 2.0 Esta opción se retiene para la compatibilidad con servicios web SOAP. No se recomienda utilizar con JSON.
- 2.1 Esta opción se retiene para la compatibilidad con servicios web SOAP. No se recomienda utilizar con JSON.
- 2.2 Esta opción se retiene para la compatibilidad con servicios web SOAP. No se recomienda utilizar con JSON.
- 3.0 El archivo de enlace de servicios web generado se despliega en una región de CICS TS 4.1 o posterior.

Nota: El soporte JSON sólo está disponible desde CICS TS 4.2 en adelante.

- 4.0 El archivo de enlace de servicios web generado se despliega correctamente en una región de CICS TS 5.2 o posterior. Con este nivel de ejecución, puede utilizar un nivel de correlación de 4.0 o anterior para el parámetro **MAPPING-LEVEL**. Puede utilizar cualquier parámetro opcional en este nivel.
- 4.1 El archivo de enlace de servicios web generado se despliega correctamente en una región de CICS TS V5.2 que tiene aplicado APAR PI67641, o en cualquier CICS TS V5.3 o región posterior. Con este nivel de ejecución, puede utilizar un nivel de correlación de 4.1 o anterior para el parámetro **MAPPING-LEVEL**.

CURRENT

El archivo de enlace de servicios web generado se despliega correctamente en una región de CICS al mismo nivel de ejecución que el que está utilizando para generar el archivo de enlace de servicio web.

PDSLIB = *valor*

Especifica el nombre del conjunto de datos particionados que contiene las estructuras de datos de lenguaje de alto nivel que se van a procesar. Los miembros del conjunto de datos que se utilizan para la solicitud y respuesta se especifican en los parámetros **REQMEM** y **RESPMEM**.

Restricción: Los registros del conjunto de datos particionados debe tener una longitud fija de 80 bytes.

PDSCP = *valor*

Especifica la página de códigos que se utiliza en los miembros del conjunto de datos particionados especificado en los parámetros **REQMEM** y **RESPMEM**, donde *value* es un número CCSID o un número de página de códigos de Java. Si no se especifica este parámetro, se utiliza la página de códigos de los servicios del sistema z/OS UNIX. Por ejemplo, puede especificar **PDSCP** = 037.

PGMINT = { **CHANNEL** | **COMMAREA** }

Para un proveedor de servicios, especifique cómo CICS pasa datos al programa de aplicación de destino:

CHANNEL

CICS utiliza una interfaz de canal para pasar datos al programa de aplicación de destino.

- En niveles de correlación anteriores a 3.0, el canal puede tener sólo un contenedor, que se utiliza para la entrada y la salida. Utilice el parámetro **CONTID** para especificar el nombre del contenedor. El nombre predeterminado es DFHWS-DATA.
- En el nivel de correlación 3.0, el canal puede contener varios contenedores. Utilice los parámetros **REQUEST-CHANNEL** y **RESPONSE-CHANNEL**. No especifique **PDSLIB** , **REQMEM** , o **RESPMEM**.

COMMAREA

CICS utiliza un área de comunicación para pasar datos al programa de aplicación de destino.

Cuando el programa de aplicación de destino ha procesado la solicitud, debe utilizar el mismo mecanismo para devolver la respuesta. Si se ha recibido la solicitud en un área de comunicación, la respuesta debe devolverse en el área de comunicación; si la solicitud se ha recibido en un contenedor, la respuesta debe devolverse en un contenedor. La longitud del contenedor o área de comunicaciones que pasa CICS al programa de aplicación de destino es mayor que las longitudes del contenedor o área de comunicación de solicitud y el contenedor o área de comunicación de la respuesta.

PGMNAME = *valor*

Especifica el nombre del programa de CICS PROGRAM para el programa de aplicación de destino que se expone como un servicio web. El soporte de servicios web de CICS enlaza con este programa.

REQMEM = *valor*

Especifica el nombre del miembro de conjunto de datos particionados que contiene la estructura de lenguaje de alto nivel para la solicitud de servicio web. Para un proveedor de servicio, la solicitud de servicio web es la entrada al programa de aplicación.

REQUEST-CHANNEL = *valor*

Especifica el nombre y la ubicación de un documento de descripción de canal. La descripción de canal describe los contenedores que puede utilizar la aplicación de proveedor de servicios web en su interfaz cuando recibe un mensaje JSON desde un solicitante de servicio web. La descripción de canal es un documento XML que debe ajustarse al esquema de canal proporcionado por CICS. Para obtener más información, consulte el apartado “Creación de un documento de descripción de canal” en la página 277.

Este parámetro sólo se puede utilizar en el nivel de correlación 3.0.

RESPMEM = *valor*

Especifica el nombre del miembro de conjunto de datos particionados que contiene la estructura de lenguaje de alto nivel para la respuesta de servicio web. Para un proveedor de servicios, la respuesta de servicio web es la salida del programa de aplicación.

RESPONSE-CHANNEL = *valor*

Especifica el nombre y la ubicación de un documento de descripción de canal. La descripción de canal describe los contenedores que puede utilizar la aplicación de proveedor de servicios web en su interfaz cuando envía un mensaje de respuesta JSON a un solicitante de servicio web. La descripción de canal es un documento XML que debe ajustarse al esquema de canal proporcionado por CICS. Para obtener más información, consulte el apartado “Creación de un documento de descripción de canal” en la página 277.

Este parámetro sólo se puede utilizar en el nivel de correlación 3.0.

STRUCTURE = (*solicitud* , *respuesta*)

Sólo para C y C++, especifica los nombres de las estructuras de alto nivel contenidas en los miembros de conjunto de datos particionados especificados en los parámetros **REQMEM** y **RESPMEM**:

solicitud

Especifica el nombre de la estructura de alto nivel que contiene la solicitud cuando se especifica el parámetro **REQMEM**. El valor predeterminado es DFHREQUEST.

El miembro de conjunto de datos particionados debe contener una estructura de alto nivel con el nombre que especifica o una estructura denominada DFHREQUEST si no especifica un nombre.

respuesta

Especifica el nombre de la estructura de alto nivel que contiene la respuesta cuando se especifica el parámetro **RESPMEM**. El valor predeterminado es DFHRESPONSE.

Si especifica un valor, el miembro de conjunto de datos particionados debe contener una estructura de alto nivel con el nombre que especifica o una estructura denominada DFHRESPONSE si no especifica un nombre.

SYNCONRETURN = { NO | **YES** }

Especifica si el servicio web remoto puede emitir un punto de sincronización.

NO El servicio web remoto no puede emitir un punto de sincronización. Este valor es el predeterminado. Si el servicio web remoto emite un punto de sincronización, falla con una terminación anómala ADPL.

YES El servicio web remoto puede emitir un punto de sincronización. Si selecciona **YES**, la tarea remota se confirma como una unidad de trabajo individual cuando se devuelve el control desde el servicio web remoto. Si el servicio web remoto actualiza un recurso recuperable y se produce un fallo después de que se devuelve, la actualización para ese recurso no se puede restituir.

TRANSACTION = *nombre*

En un proveedor de servicios, este parámetro especifica de uno a cuatro caracteres del nombre de una transacción de alias que puede iniciar la interconexión. El valor de este parámetro se utiliza para definir el atributo **TRANSACTION** del recurso URIMAP cuando se crea automáticamente utilizando el mandato de exploración **PIPELINE**.

Caracteres aceptables:

A-Z a-z 0-9 \$ @ # _ < >

TRUNCATE-NULL-ARRAYS = { **DISABLED** | **ENABLED** }

Especifica cómo se procesan las matrices estructuradas en el nivel de correlación 4.1 o superior. Si está habilitado, CICS intentará reconocer los registros vacíos dentro de una matriz (consulte **TRUNCATE-NULL-ARRAY-VALUES** para obtener más información sobre la identificación de registros vacíos). Si se detectan cinco registros de matriz vacíos consecutivos, la matriz se trunca en el primero de esos registros cuando se genera XML/JSON. Esta prestación de recorte sólo se habilita para matrices con contenido estructurado, las matrices de campos primitivos simples no están sujetos al recorte. El recorte de matrices puede dar como resultado una representación más precisa de los datos en JSON/XML, pero no sin riesgo. Si cinco registros de datos consecutivos se identifican de forma incorrecta como almacenamiento no

inicializado (quizás porque contienen legítimamente valores bajos), se puede experimentar una pérdida de datos. Si TRUNCATE-NULL-ARRAYS está habilitado y TRUNCATE-NULL-ARRAY-VALUES no se ha establecido, se utiliza el valor predeterminado para TRUNCATE-NULL-ARRAY-VALUES.

TRUNCATE-NULL-ARRAY-VALUES = { NULL | **SPACE** | **ZERO** }

Especifica qué valores se tratan como vacíos para el proceso de TRUNCATE-NULL-ARRAYS en el nivel de correlación 4.1 o superior. De forma predeterminada, el valor nulo (0x00, o los valores bajos) se tratan como vacíos. Si todos los bytes de almacenamiento dentro de un registro de una matriz estructurada contienen nulos, entonces todo el registro se considera vacío. Se pueden especificar uno o más de los valores NULL, SPACE y ZERO en una lista separada por comas, donde NULL implica un carácter nulo (0x00), SPACE implica un SBCS EBCDIC Space (0x40) y ZERO implica un cero decimal con zona (0xF0). Cualquier combinación coincidente de los bytes seleccionados dentro un registro de matriz estructurado provocará que todo el registro se identifique como vacío. Si TRUNCATE-NULL-ARRAY-VALUES tiene un valor definido, entonces TRUNCATE-NULL-ARRAYS debe estar habilitado.

URI = *valor*

Este parámetro especifica el URI absoluto o relativo que utiliza un cliente para acceder al servicio web. CICS utiliza el valor que se especifica cuando genera un recurso URIMAP desde un archivo de enlace de servicio web que ha creado DFHLS2JS. El parámetro especifica el componente de vía de acceso del URI al que se aplica la definición URIMAP.

USERID = *id*

En un proveedor de servicios, este parámetro especifica de uno a ocho caracteres del ID de usuario, que puede utilizar cualquier cliente web. Para una respuesta generada por la aplicación o un servicio web, la transacción de alias se conecta bajo este ID de usuario. El valor de este parámetro se utiliza para definir el atributo USERID del recurso URIMAP cuando se crea automáticamente utilizando el mandato de exploración **PIPELINE**.

Caracteres aceptables:

A-Z a-z 0-9 \$ @ #

WSBIND = *valor*

El nombre completo del archivo de enlace de servicio web de z/OS UNIX. DFHLS2JS crea el archivo, pero no la estructura de directorio, si no existe. La extensión de archivo es .wsbind.

Otra información

- El ID de usuario que utiliza DFHLS2JS para ejecutar debe configurarse para utilizar los servicios del sistema UNIX. El ID de usuario debe tener permiso de lectura para las bibliotecas PDS y la estructura de archivo de CICS z/OS UNIX y permiso de escritura para los directorios especificados en los parámetros **LOGFILE**, **WSBIND** y **JSON Schema**.
- El ID de usuario debe tener una asignación de almacenamiento lo suficientemente grande para ejecutar Java.
- El JCL tiene una longitud de parámetro máxima de 100 caracteres. Esta longitud de parámetro se puede aumentar utilizando la sentencia **STDPARM**, para obtener más información, consulte *z/OS UNIX System Services User Guide*.

Ejemplo

```
//LS2JS JOB '  
información de contabilidad  
'  
,  
name,MSGCLASS=A  
// SET QT='''  
//JAVAPROG EXEC DFHLS2JS,  
// TMPFILE=&QT.&SYSUID.&QT,  
//INPUT.SYSUT1 DD *  
PDSLIB=CICSHLQ.SDFHSAMP  
REQMEM=DFH0XCP4  
RESPMEM=DFH0XCP4  
JSON-SCHEMA-REQUEST=/u/exampleapp/json/example_request.json  
JSON-SCHEMA-RESPONSE=/u/exampleapp/json/example_response.json  
LANG=COBOL  
LOGFILE=/u/exampleapp/wsbinding/example.log  
MAPPING-LEVEL=4.0  
CHAR-VARYING=COLLAPSE  
PGMNAME=DFH0XCMN  
URI=http://myserver.example.org:8080/exampleApp/example  
PGMINT=COMMAREA  
SYNCONRETURN=YES  
WSBIND=/u/exampleapp/wsbinding/example.wsbinding  
/*
```

DFHJS2LS: Conversión de esquema JSON a un lenguaje de alto nivel para servicios solicitud-respuesta

El procedimiento DFHJS2LS genera una estructura de datos de lenguaje de alto nivel y un archivo de enlace de servicio web a partir de un esquema JSON. Puede utilizar DFHJS2LS cuando se prepara para crear un programa de aplicación CICS como proveedor de servicios. Este tema lista las sentencias de control de trabajo, parámetros simbólicos, parámetros de entrada y sus descripciones para DFHJS2LS.

El procedimiento DFHJS2LS JCL está instalado en el conjunto de datos *HLQ* .XDFHINST , donde *HLQ* es el calificador de alto nivel donde está instalado CICS.

Sentencias de control de trabajos para DFHJS2LS

JOB Inicia el trabajo.

EXEC Especifica el nombre del procedimiento (DFHJS2LS).

INPUT.SYSUT1 DD

Especifica la entrada. Los parámetros de entrada se especifican normalmente en el flujo de entrada. Sin embargo, se pueden definir en un conjunto de datos en miembro de un conjunto de datos particionados.

Parámetros simbólicos

Los siguientes parámetros simbólicos se definen en DFHJS2LS:

JAVADIR = *vía de acceso*

Especifica el nombre del directorio Java que utiliza DFHJS2LS. El valor de este parámetro se añade a */usr/lpp/* para crear un nombre de vía de acceso completo de */usr/lpp/ vía de acceso* .

Generalmente, no es necesario especificar este parámetro. El valor predeterminado es el valor proporcionado para el trabajo de instalación (DFHISTAR) de CICS en el parámetro **JAVADIR**.

PATHPREFIX = *prefijo*

Especifica un prefijo que amplía la vía de acceso del directorio z/OS UNIX que se utiliza en otros parámetros o '' (serie vacía) si no se utiliza un prefijo.

Generalmente, no es necesario especificar este parámetro. El valor predeterminado es el valor proporcionado para el trabajo de instalación (DFHISTAR) de CICS en el parámetro **PATHPREF**.

SERVICE = *valor*

Utilice este parámetro sólo cuando el soporte de IBM le indique que lo haga.

TMPDIR = *tmpdir*

Especifica la ubicación de un directorio en z/OS UNIX que DFHJS2LS utiliza como espacio de trabajo temporal. El ID de usuario bajo el que se ejecuta el trabajo debe tener permiso de lectura y escritura para este directorio.

El valor predeterminado es /tmp.

TMPFILE = *tmpprefix*

Especifica un prefijo que DFHJS2LS utiliza para construir los nombres de los archivos de espacio de trabajo temporal.

El valor predeterminado es JS2LS.

USSDIR = *vía de acceso*

Especifica el nombre del directorio CICS TS en el sistema de archivos de los servicios del sistema UNIX. El valor de este parámetro se añade a /usr/lpp/cicsts/ para crear un nombre de vía de acceso completo de /usr/lpp/cicsts/ *vía de acceso* . Esto debe especificarse como '.' (punto) si se utiliza el valor predeterminado.

Generalmente, no es necesario especificar este parámetro. El valor predeterminado es el valor proporcionado para el trabajo de instalación (DFHISTAR) de CICS en el parámetro **USSDIR**.

El espacio de trabajo temporal

DFHJS2LS crea estos tres archivos temporales en tiempo de ejecución:

```
tmpdir / tmpprefix .in  
tmpdir / tmpprefix .out  
tmpdir / tmpprefix .err
```

donde:

tmpdir es el valor que se especifica en el parámetro **TMPDIR**.

tmpprefix es el valor que se especifica en el parámetro **TMPFILE**.

Los nombres predeterminados para los archivos, cuando no se especifica **TMPDIR** y **TMPFILE** son los siguientes:

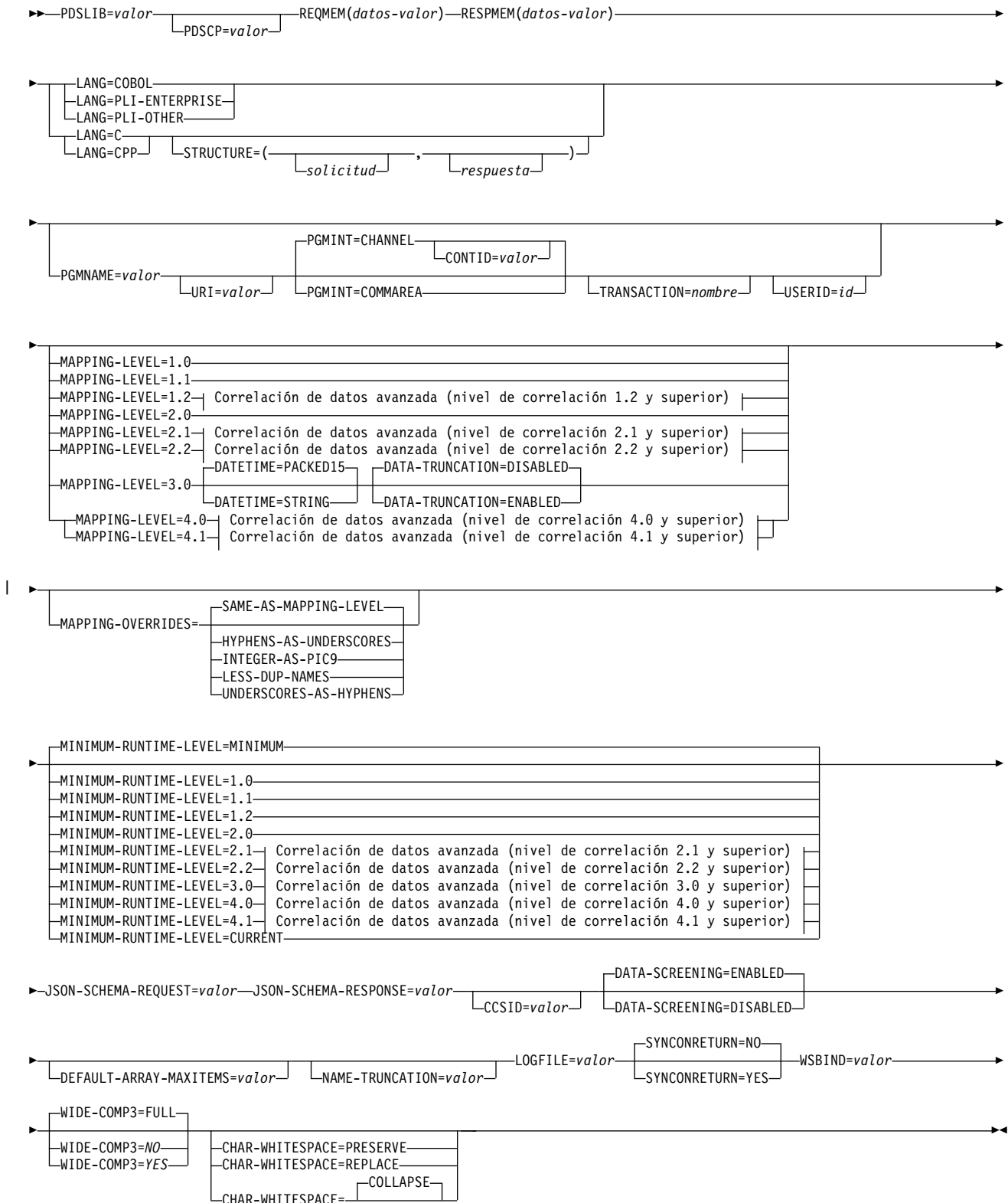
```
/tmp/JS2LS.in  
/tmp/JS2LS.out  
/tmp/JS2LS.err
```

Importante: DFHJS2LS no bloquea el acceso a los archivos z/OS UNIX o los miembros del conjunto de datos. Por lo tanto, si dos o más instancias de DFHJS2LS se ejecutan de forma simultánea y utilizan los mismos archivos de espacio de trabajo temporal, nada impide a un trabajo sobrescribir los archivos de espacio de trabajo mientras otro trabajo los está utilizando, llevando a errores imprevisibles.

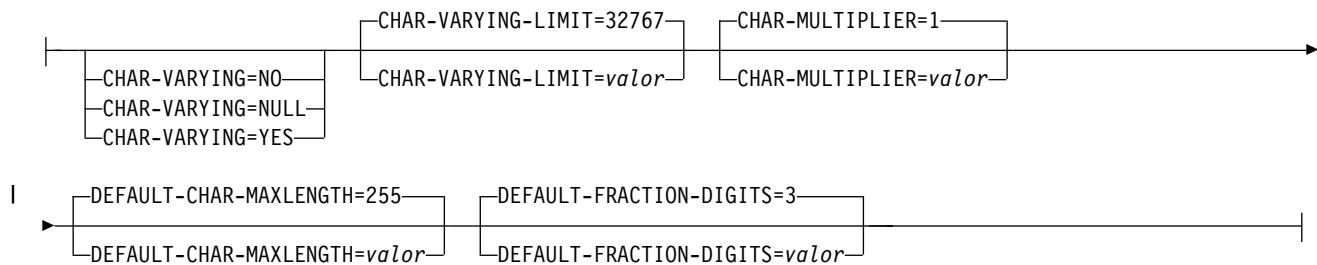
Por lo tanto, se aconseja crear un convenio de denominación y procedimientos operativos que eviten esta situación. Por ejemplo, puede utilizar el parámetro simbólico de sistema **SYSUID** para generar nombres de archivo de espacios de

trabajo que son exclusivos para un usuario individual. Estos archivos temporales se suprimen antes de la finalización del trabajo.

Parámetros de entrada de DFHJS2LS



Correlación de datos avanzada (nivel de correlación 1.2 y superior):



Correlación de datos avanzada (nivel de correlación 2.1 y superior):



Utilización de parámetros

- Puede especificar los parámetros de entrada en cualquier orden.
- Cada parámetro se debe iniciar en una nueva línea.
- Un parámetro, y su carácter de continuación, si utiliza uno, no deben sobrepasar la columna 72; las columnas 73 a 80 deben contener espacios en blanco.
- Si un parámetro es demasiado largo para ajustarse en una única línea, utilice el carácter de asterisco (*) al final de la línea para indicar que el parámetro continúa en la línea siguiente. Todo, incluidos los espacios antes del asterisco, se considera parte del parámetro. Por ejemplo:

```
WSBIND=wsbinddir*  
      /app1
```

es equivalente a

```
WSBIND=wsbinddir/app1
```

- Un carácter # en la primera posición de carácter de la línea es un carácter de comentario. La línea se ignora.
- Una coma en la última posición de carácter de la línea es un separador de línea opcional y se ignora.

Descripciones de parámetros

CCSID = *valor*

Especifica el CCSID que se utiliza en tiempo de ejecución para codificar datos de caracteres en la estructura de datos de la aplicación. El valor de este parámetro sustituye el valor del parámetro de inicialización del sistema **LOCALCCSID**. El *valor* debe ser un EBCDIC CCSID soportado por Java y . Si no especifica este parámetro, la estructura de datos de aplicación se codifica utilizando el CCSID especificado en el parámetro de inicialización del sistema.

CHAR-MULTIPLIER = { 1 | *valor* }

Especifica el número de bytes que se permiten para cada carácter cuando el nivel de correlación es 1.2 o posterior. El *valor* de este parámetro puede ser un entero positivo en el rango de 1 a 2.147.483.647. Todas las correlaciones basadas en caracteres no numéricos están sujetas a este multiplicador. Los campos binario, numérico, con zona y decimal empaquetado no están sujetos a este multiplicador.

Este parámetro puede ser útil si, por ejemplo, está planificando utilizar caracteres DBCS donde puede optar por un multiplicador de 3 para permitir espacio para caracteres de cambio y desplazamiento potenciales alrededor de todos los caracteres de doble byte en tiempo de ejecución.

Cuando establece **CCSID=1200** (indicando UTF-16), los únicos valores válidos para **CHAR-MULTIPLIER** son 2 o 4 . Cuando utiliza UTF-16, el valor predeterminado es 2 . Utilice **CHAR-MULTIPLIER=2** cuando espera que los datos de aplicación contengan caracteres que requieren 1 unidad de codificación UTF-16. Utilice **CHAR-MULTIPLIER=4** cuando espera que los datos de aplicación contengan caracteres que requieren 2 unidades de codificación UTF-16.

Nota: Establecer **CHAR-MULTIPLIER** en 1 no excluye el uso de caracteres DBCS, y establecerlo en 2 no excluye el uso de pares de sustitución UTF-16. Sin embargo, si los caracteres amplios utilizan de forma rutinaria algunos valores válidos no se ajustarán al campo asignado. Si se utiliza un valor **CHAR-MULTIPLIER** mayor, es posible almacenar más caracteres en el campo asignado que los que son válidos en el XML. Se debe tener cuidado para ajustarse a las restricciones de rango adecuadas.

CHAR-VARYING = { **NO** | **NULL** | **YES** }

Especifica cómo se correlacionan datos de caracteres de longitud variable cuando el nivel de correlación es 1.2 o superior. Los tipos de datos binarios de longitud variable se correlacionan siempre con un contenedor o una estructura variable. Si no especifica este parámetro, la correlación predeterminada depende del lenguaje especificado. Puede seleccionar estas opciones:

NO Los datos de caracteres de longitud variable se correlacionan como series de longitud fija.

NULL Los datos de caracteres de longitud variable se correlacionan con series terminadas en nulo.

YES Los datos de caracteres de longitud variable se correlacionan con un tipo de datos CHAR VARYING en PL/I. En los lenguajes COBOL, C y C++, los datos de caracteres de longitud variable se correlacionan con una representación equivalente que consta de dos elementos relacionados: la longitud de datos y los datos.

CHAR-VARYING-LIMIT = { **32767** | *valor* }

Especifica el tamaño máximo de datos binarios y de datos de caracteres de longitud variable que se correlacionan con la estructura de lenguaje cuando el nivel de correlación es 1.2 o superior. Si los datos binarios o de caracteres son mayores que el valor especificado en este parámetro, se correlacionan con un contenedor y el nombre de contenedor se utiliza en la estructura de lenguaje generada. El valor puede estar comprendido entre 0 y el valor predeterminado 32.767 bytes.

CHAR-WHITESPACE = **COLLAPSE** | **REPLACE** | **PRESERVE**

Especifica cómo CICS maneja el espacio en blanco en valores de tipo cadena.

COLLAPSE

Los espacios en blanco de inicio, final e incluidos se eliminan y todas las pestañas, líneas nuevas y espacios consecutivos se sustituyen por caracteres de un único espacio.

REPLACE

Cualquier pestaña o línea nueva se sustituye por el número de espacios adecuado.

PRESERVE

Retiene cualquier espacio en blanco en el valor de datos.

Si no se establece el parámetro **CHAR-WHITESPACE**, el espacio en blanco se contrae.

Nota: Este parámetro no se aplica a campos con un formato de fecha-hora, uri, base64Binary o hexBinary, donde el espacio en blanco siempre se contrae.

CONTID = *valor*

En un proveedor de servicios, especifique el nombre del contenedor que tiene la estructura de datos de nivel superior utilizada para representar un mensaje JSON.

La longitud del contenedor que CICS pasa al programa de aplicación de destino es mayor que las longitudes del contenedor de solicitud y el contenedor de respuesta.

DATA-SCREENING = { **ENABLED** | **DISABLED** }

Especifica si los datos proporcionados por la aplicación se analizan para detectar errores.

ENABLED

Los datos de tiempo de ejecución proporcionados por la aplicación que son incoherentes con la estructura de lenguaje, se tratan como un error y se emite un mensaje DFHPI1010. Se devuelve una respuesta de error a la aplicación.

DISABLED

Los valores de los datos de tiempo de ejecución proporcionados por la aplicación que son incoherentes con la estructura de lenguaje se sustituyen por valores predeterminados. Por ejemplo, un cero sustituye un valor erróneo en un campo numérico. No se emite mensaje DFHPI1010 y se devuelve una respuesta normal a la aplicación. Esta característica se puede utilizar para evitar las respuestas de error INVALID_PACKED_DEC y INVALID_ZONED_DEC generadas desde los campos de salida no inicializados.

DATA-TRUNCATION = { **DISABLED** | **ENABLED** }

Especifica si los datos de longitud variable se toleran en una estructura de campo de longitud fija:

DISABLED

Si los datos son menores que la longitud fija que está esperando CICS, CICS rechaza los datos truncados y emite un mensaje de error.

ENABLED

Si los datos son menores que la longitud fija que está esperando CICS, CICS tolera los datos truncados y procesa los datos que faltan como valores nulos.

DATETIME = { **PACKED15** | **STRING** }

Especifica cómo se correlacionan los elementos de fecha y hora JSON con la estructura de lenguaje.

PACKED15

El valor predeterminado es que cualquier elemento de fecha y hora JSON se procesa como una indicación de fecha y hora y se correlaciona con el formato ABSTIME de CICS.

STRING

El elemento de fecha y hora de JSON se procesa como texto.

DEFAULT-ARRAY-MAXITEMS = *valor*

Especifica el límite de matriz máximo a aplicar donde no se implica información de aparición máxima (maxItems) en el esquema JSON. Si no se establece este parámetro, no se aplica límite máximo. El valor de este parámetro puede ser un entero positivo en el rango de 1 a 2147483647. Este parámetro se puede combinar con el uso del parámetro **INLINE-MAXOCCURS-LIMIT** para que influya en la forma en la que se correlacionan las matrices JSON en las estructuras de lenguaje.

DEFAULT-CHAR-MAXLENGTH = { 255 | *valor* }

Especifica la longitud de matriz predeterminada de los datos de caracteres en caracteres para correlaciones donde no está implicada la longitud en el documento de descripción servicio web, cuando el nivel de correlación es 1.2 o posterior. El valor de este parámetro puede ser un entero positivo en el rango de 1 a 2.147.483.647.

DEFAULT-FRACTION-DIGITS = { 3 | *valor* }

Especifica el número de dígitos de fracción predeterminado a utilizar en un tipo de esquema decimal XML. El valor predeterminado es 3. Para COBOL, el rango válido es 0-17, o 0-30 si se está utilizando **WIDE-COMP3**. Para C o PLI, el rango válido es 0-30.

INLINE-MAXOCCURS-LIMIT = { 1 | *valor* }

Especifica si se utiliza el contenido de repetición variable en línea basado en la palabra clave de esquema JSON maxItems. El contenido de repetición variable que se correlaciona en línea se coloca en el contenedor actual con el resto de la estructura de lenguaje generada. El contenido de repetición variable se almacena en dos partes, como contador que almacena el número de apariciones de los datos y como una matriz que almacena cada aparición de los datos. La correlación alternativa para el contenido de repetición variable es una correlación basada en contenido, que almacena el número de apariciones de los datos y el nombre del contenedor donde se colocan los datos. Almacenar los datos en un contenedor distinto tiene implicaciones de rendimiento que pueden hacer que sea preferible la correlación en línea.

El parámetro **INLINE-MAXOCCURS-LIMIT** está disponible sólo del nivel de correlación 2.1 en adelante. El valor de **INLINE-MAXOCCURS-LIMIT** puede ser un entero positivo en el rango de 0 a 32.767. Un valor de 0 indica que no se utiliza la correlación en línea. Un valor de 1 garantiza que los elementos opcionales se correlacionan en línea. Si el *valor* del atributo maxOccurs es mayor del *valor* de **INLINE-MAXOCCURS-LIMIT**, se utiliza la correlación basada en contenedor; de lo contrario, se utiliza la correlación en línea.

Cuando decide si desea que las listas de repetición variable se correlacionen en línea, tenga en cuenta la longitud de un único elemento de datos recurrentes. Si se producen muchas instancias de longitud larga, es preferible la correlación basada en contenedor; si se producen muchas instancias de longitud corta, es preferible la correlación en línea.

JSON-SCHEMA-REQUEST = *valor*

Este es un parámetro obligatorio.

El valor indica la ubicación de los servicios del sistema de UNIX donde se almacena el esquema JSON de solicitud.

JSON-SCHEMA-RESPONSE = *valor*

Este es un parámetro obligatorio.

El valor indica la ubicación de los servicios del sistema de UNIX donde se almacena el esquema JSON de respuesta.

LANG = COBOL

Especifica que el lenguaje de programación de la estructura de lenguaje de alto nivel es COBOL.

LANG = PLI-ENTERPRISE

Especifica que el lenguaje de programación de la estructura de lenguaje de alto nivel es Enterprise PL/I.

LANG = PLI-OTHER

Especifica que el lenguaje de programación de la estructura de lenguaje de alto nivel es un nivel de PL/I distinto a Enterprise PL/I.

LANG = C

Especifica que el lenguaje de programación de la estructura de lenguaje de alto nivel es C.

LANG = CPP

Especifica que el lenguaje de programación de la estructura de lenguaje de alto nivel es C++.

LOGFILE = *valor*

El nombre completo del archivo de z/OS UNIX en el que DFHJS2LS escribe la información de rastreo y registro de actividad. DFHJS2LS crea el archivo, pero no la estructura de directorio, si no existe.

Normalmente, no utiliza este archivo, pero puede solicitarlo la organización de servicio de IBM si encuentra problemas con DFHJS2LS.

MAPPING-LEVEL = { 1.0 | 1.1 | 1.2 | 2.0 | 2.1 | 2.2 | 3.0 | 4.0 | 4.1 }

Especifica el nivel de correlación que utiliza DFHJS2LS cuando genera el archivo de enlace de servicio web y la estructura de lenguaje. Puede seleccionar estas opciones:

- 1.0** Esta opción se retiene para la compatibilidad con servicios web SOAP. No se recomienda utilizar con JSON.
- 1.1** Esta opción se retiene para la compatibilidad con servicios web SOAP. No se recomienda utilizar con JSON.
- 1.2** Esta opción se retiene para la compatibilidad con servicios web SOAP. No se recomienda utilizar con JSON.
- 2.0** Esta opción se retiene para la compatibilidad con servicios web SOAP. No se recomienda utilizar con JSON.
- 2.1** Esta opción se retiene para la compatibilidad con servicios web SOAP. No se recomienda utilizar con JSON.
- 2.2** Esta opción se retiene para la compatibilidad con servicios web SOAP. No se recomienda utilizar con JSON.
- 3.0** Utilice este nivel de correlación para generar un esquema JSON utilizando el conjunto de opciones completo disponible.
- 4.0** Utilice este nivel de correlación con una región de CICS TS 5.2 cuando dese utilizar UTF-16.
- 4.1** Para un soporte de matriz truncable, utilice este nivel de correlación con una región de CICS V5.2 que tiene aplicado APAR PI67641, o cualquier región de CICS V5.3 o posterior.

Para obtener más información sobre los niveles de correlación, consulte Mapping levels for the CICS assistants.

MAPPING-OVERRIDES = { SAME-AS-MAPPING-LEVEL | HYPHENS-AS-UNDERScores | INTEGER-AS-PIC9 | LESS-DUP-NAMES | UNDERScores-AS-HYPHENS }

Especifica si el comportamiento predeterminado se sustituye para el nivel de correlación especificado cuando se generan estructuras de lenguaje.

SAME-AS-MAPPING-LEVEL

Este parámetro genera estructuras de lenguaje en el mismo estilo que el nivel de correlación. Es el valor predeterminado.

HYPHENS-AS-UNDERScores

Sólo para PL/I. Este parámetro convierte cualquier guión del esquema JSON en guiones bajos en lugar del carácter X, para mejorar la legibilidad de las estructuras de lenguaje PL/I generadas. Para obtener más información, consulte el apartado JSON schema to PL/I mapping.

INTEGER-AS-PIC9

Únicamente para COBOL y DFHJS2LS. Este parámetro genera estructuras de lenguaje que contienen valores enteros del esquema JSON como numerales en lugar de caracteres alfanuméricos.

LESS-DUP-NAMES

Este parámetro genera nombres de campo de estructura no estructural con `_value` al final del nombre para habilitar la referencia directa al campo. Por ejemplo, en la siguiente estructura PLI, cuando se especifica `MAPPING-OVERRIDES=LESS-DUP-NAMES`, el campo de nivel 12 `streetName` lleva el sufijo `_value` :

```
09 streetName,
12 streetName CHAR(255) VARYING
UNALIGNED,
12 filler BIT (7),
12 attr_nil_streetName_value BIT (1),
```

La estructura resultante es la siguiente:

```
09 streetName,
12 streetName_value CHAR(255) VARYING
UNALIGNED,
12 filler BIT (7),
12 attr_nil_streetName_value BIT (1),
```

UNDERScores-AS-HYPHENS

Esta opción está habilitada automáticamente en el nivel de correlación 4.0.

Solo para COBOL. Este parámetro convierte cualquier guión bajo en esquema JSON en guiones, en lugar del carácter X, para mejorar la legibilidad de las estructuras de lenguaje COBOL generadas. Si se produce un conflicto de nombres de campo, los campos se numeran para garantizar que con exclusivos. Para obtener más información, consulte el apartado JSON schema to COBOL mapping.

MINIMUM-RUNTIME-LEVEL = { MINIMUM | 1.0 | 1.1 | 1.2 | 2.0 | 2.1 | 2.2 | 3.0 | 4.0 | 4.1 | CURRENT }

Especifica el entorno de ejecución de CICS mínimo en el que se puede desplegar el archivo de enlace de servicio web. Si selecciona un nivel que no coincide con los demás parámetros que ha especificado, recibe un mensaje de error. Puede seleccionar estas opciones:

MINIMUM

El nivel de tiempo de ejecución más bajo posible de CICS se asigna automáticamente a los parámetros determinados que seleccione.

- 1.0 Esta opción se retiene para la compatibilidad con servicios web SOAP. No se recomienda utilizar con JSON.
- 1.1 Esta opción se retiene para la compatibilidad con servicios web SOAP. No se recomienda utilizar con JSON.
- 1.2 Esta opción se retiene para la compatibilidad con servicios web SOAP. No se recomienda utilizar con JSON.
- 2.0 Esta opción se retiene para la compatibilidad con servicios web SOAP. No se recomienda utilizar con JSON.
- 2.1 Esta opción se retiene para la compatibilidad con servicios web SOAP. No se recomienda utilizar con JSON.
- 2.2 Esta opción se retiene para la compatibilidad con servicios web SOAP. No se recomienda utilizar con JSON.
- 3.0 El archivo de enlace de servicios web generado se despliega en una región de CICS TS 4.1 o posterior.

Nota: El soporte JSON sólo está disponible desde CICS TS 4.2 en adelante.

- 4.0 El archivo de enlace de servicios web generado se despliega correctamente en una región de CICS TS 5.2 o posterior. Con este nivel de ejecución, puede utilizar un nivel de correlación de 4.0 o anterior para el parámetro **MAPPING-LEVEL**. Puede utilizar cualquier parámetro opcional en este nivel.
- 4.1 El archivo de enlace de servicios web generado se despliega correctamente en una región de CICS TS V5.2 que tiene aplicado APAR PI67641, o en cualquier CICS TS V5.3 o región posterior. Con este nivel de ejecución, puede utilizar un nivel de correlación de 4.1 o anterior para el parámetro **MAPPING-LEVEL**.

CURRENT

El archivo de enlace de servicios web generado se despliega correctamente en una región de CICS al mismo nivel de ejecución que el que está utilizando para generar el archivo de enlace de servicio web.

NAME-TRUNCATION = { **LEFT** | **RIGHT** }

Especifica si los nombres JSON están trucados desde la izquierda o la derecha. El asistente de servicios web de CICS trunca nombres JSON en la longitud adecuada para el lenguaje de alto nivel especificado; de forma predeterminada los nombres se truncan desde la derecha.

PDSCP = *valor*

Especifica la página de códigos que se utiliza en los miembros del conjunto de datos particionados especificado en los parámetros **REQMEM** y **RESPMEM**, donde *value* es un número CCSID o un número de página de códigos de Java. Si no se especifica este parámetro, se utiliza la página de códigos de los servicios del sistema z/OS UNIX. Por ejemplo, puede especificar **PDSCP** = 037.

PDSLIB = *valor*

Especifica el nombre del conjunto de datos particionados que contiene el lenguaje de alto nivel generado. Los miembros del conjunto de datos que se utilizan para la solicitud y respuesta se especifican en los parámetros **REQMEM** y **RESPMEM**.

PGMINT = { **CHANNEL** | **COMMAREA** }

Para un proveedor de servicios, especifique cómo CICS pasa datos al programa de aplicación de destino:

CHANNEL

CICS utiliza una interfaz de canal para pasar datos al programa de aplicación de destino.

COMMAREA

CICS utiliza un área de comunicación para pasar datos al programa de aplicación de destino.

Cuando el programa de aplicación de destino ha procesado la solicitud, debe utilizar el mismo mecanismo para devolver la respuesta. Si se ha recibido la solicitud en un área de comunicación, la respuesta debe devolverse en el área de comunicación; si la solicitud se ha recibido en un contenedor, la respuesta debe devolverse en un contenedor. La longitud del contenedor o área de comunicaciones que pasa CICS al programa de aplicación de destino es mayor que las longitudes del contenedor o área de comunicación de solicitud y el contenedor o área de comunicación de la respuesta.

PGMNAME = *valor*

Especifica el nombre de un recurso PROGRAM de CICS.

Cuando se utiliza DFHJS2LS para generar un archivo de enlace de servicio web que se utiliza en un proveedor de servicios, debe proporcionar este parámetro. Especifica el nombre de recurso del programa de aplicación que se expone como un servicio web.

Cuando se utiliza DFHJS2LS para generar un archivo de enlace de servicio web que se utiliza en un solicitante de servicio, omita este parámetro.

REQMEM = *valor*

Especifica un prefijo de uno a seis caracteres que utiliza DFHJS2LS para generar los nombres de los miembros de conjunto de datos particionados que contiene las estructuras de lenguaje de alto nivel para la solicitud de servicio web, que son los datos de entrada para el programa de aplicación.

DFHJS2LS genera el nombre del miembro de conjunto de datos particionados añadiendo 01 al prefijo.

RESPMEM = *valor*

Especifica un prefijo de uno a seis caracteres que utiliza DFHJS2LS para generar los nombres de los miembros de conjunto de datos particionados que contiene las estructuras de lenguaje de alto nivel para la respuesta de servicio web, que son los datos de salida para el programa de aplicación.

DFHJS2LS genera el nombre del miembro de conjunto de datos particionados añadiendo 01 al prefijo.

STRUCTURE = (*solicitud* , *respuesta*)

Sólo para C y C++, especifique cómo se generan los nombres de las estructuras de solicitud y respuesta.

Las estructuras de solicitud y respuesta generadas reciben los nombres de *solicitud01* y *respuesta01*.

Si se omiten uno o ambos nombres, las estructuras tienen el mismo nombre que los nombres de miembro de conjunto de datos particionados generados desde los parámetros **REQMEM** y **RESPMEM** que especifica.

SYNCONRETURN = { **NO** | **YES** }

Especifica si el servicio web remoto puede emitir un punto de sincronización.

- NO** El servicio web remoto no puede emitir un punto de sincronización. Este valor es el predeterminado. Si el servicio web remoto emite un punto de sincronización, falla con una terminación anómala ADPL.
- YES** El servicio web remoto puede emitir un punto de sincronización. Si selecciona YES , la tarea remota se confirma como una unidad de trabajo individual cuando se devuelve el control desde el servicio web remoto. Si el servicio web remoto actualiza un recurso recuperable y se produce un fallo después de que se devuelve, la actualización para ese recurso no se puede restituir.

TRANSACTION = *nombre*

En un proveedor de servicios, este parámetro especifica de uno a cuatro caracteres del nombre de una transacción de alias que puede iniciar la interconexión. El valor de este parámetro se utiliza para definir el atributo TRANSACTION del recurso URIMAP cuando se crea automáticamente utilizando el mandato de exploración **PIPELINE**.

Caracteres aceptables:

A-Z a-z 0-9 \$ @ # _ < >

URI = *valor*

En un proveedor de servicios, este parámetro especifica el URI relativo que utiliza un cliente para acceder al servicio web. CICS utiliza el valor que se especifica cuando genera un recurso URIMAP desde un archivo de enlace de servicio web que ha creado DFHJS2LS. El parámetro especifica el componente de vía de acceso del URI al que se aplica la definición URIMAP.

USERID = *id*

En un proveedor de servicios, este parámetro especifica de uno a ocho caracteres del ID de usuario, que puede utilizar cualquier cliente web. Para una respuesta generada por la aplicación o un servicio web, la transacción de alias se conecta bajo este ID de usuario. El valor de este parámetro se utiliza para definir el atributo USERID del recurso URIMAP cuando se crea automáticamente utilizando el mandato de exploración **PIPELINE**.

Caracteres aceptables:

A-Z a-z 0-9 \$ @ #

WIDE-COMP3 = { FULL | NO | YES }

Controla el tamaño máximo de la longitud variable decimal empaquetada en la estructura de lenguaje COBOL o PL/I generada.

FULL Para COBOL y PL/I. DFHJS2LS genera un campo decimal empaquetado que es lo suficientemente grande como para contener todos los valores válidos. El tamaño máximo es de 31 dígitos. Es el valor predeterminado.

NO Solo para COBOL. DFHJS2LS limita la longitud variable de decimal empaquetado a 10 al generar el tipo de estructura de lenguaje COBOL COMP-3. Si el tamaño del decimal empaquetado es mayor de 18, se emite un mensaje DFHPI9022W para indicar que el tipo especificado se está restringiendo a un total de 18 dígitos.

YES Solo para COBOL. DFHJS2LS soporta el tamaño máximo de 31 al generar el tipo de estructura de lenguaje COBOL COMP-3.

Nota: Las opciones NO y YES generan campos que no pueden representar todos los valores válidos; la opción FULL evita este problema. Sin embargo, la opción FULL no permite que algunos valores no válidos se representen en el campo de decimal empaquetado. Por ejemplo, si un esquema indica que hay un máximo de cinco dígitos y un máximo de dos dígitos fraccionales, la opción FULL generará un campo de decimal empaquetado que permite siete dígitos, y esto permite espacio para valores válidos como 25000 y 999,99, pero también proporciona espacio para algunos valores no válidos como 9999,99. Cuando utilice la opción FULL, tenga cuidado de no generar valores no válidos en datos de aplicación.

WSBIND = *valor*

El nombre completo del archivo de enlace de servicio web de z/OS UNIX. DFHJS2LS crea el archivo, pero no la estructura de directorio, si no existe. La extensión de archivo predeterminada es .wsbind.

Otra información

- El ID de usuario bajo el que se ejecuta DFHJS2LS debe configurarse para utilizar los servicios de sistema UNIX. El ID de usuario debe tener permiso de lectura para la estructura de archivo de CICS z/OS UNIX y las bibliotecas PDS y permiso de escritura para los directorio especificados en los parámetros **LOGFILE**, **WSBIND** y **WSDL**.
- El ID de usuario debe tener una asignación de almacenamiento lo suficientemente grande para ejecutar Java.
- El JCL tiene una longitud de parámetro máxima de 100 caracteres. Esto puede aumentarse utilizando la sentencia **STDPARM**. Para obtener más información, consulte el apartado *z/OS UNIX System Services User Guide*.

Ejemplo

```
//JS2LS JOB '
información de contabilidad
',
name,MSGCLASS=A
// SET QT='''
//JAVAPROG EXEC DFHJS2LS,
// TMPFILE=&QT.&SYSUID.&QT
/INPUT.SYSUT1 DD *
PDSLIB=//CICSHLQ.SDFHSAMP
REQMEM=CPYBK1
RESPMEM=CPYBK2
JSON-SCHEMA-REQUEST=example.json
JSON-SCHEMA-RESPONSE=example.json
LANG=COBOL
LOGFILE=/u/exampleapp/wsbind/example.log
MAPPING-LEVEL=4.0
CHAR-VARYING=NULL
INLINE-MAXOCCURS-LIMIT=2
PGMNAME=DFH0XCMN
URI=exampleApp/example
PGMINT=COMMAREA
SYNCONRETURN=YES
WSBIND=/u/exampleapp/wsbind/example.wsbind
/*
```

DFHJS2LS: Conversión de esquema JSON a un lenguaje de alto nivel para servicios RESTful

El procedimiento DFHJS2LS genera una estructura de datos de lenguaje de alto nivel y un archivo de enlace de servicio web a partir de un esquema JSON. Puede utilizar DFHJS2LS cuando se prepare para crear un proveedor de servicios JSON

de RESTful. Este tema lista las sentencias de control de trabajo, parámetros simbólicos, parámetros de entrada y sus descripciones DFHJS2LS.

El procedimiento DFHJS2LS JCL está instalado en el conjunto de datos *HLQ* .XDFHINST , donde *HLQ* es el calificador de alto nivel donde está instalado CICS.

Sentencias de control de trabajos para DFHJS2LS

JOB Inicia el trabajo.

EXEC Especifica el nombre del procedimiento (DFHJS2LS).

INPUT.SYSUT1 DD

Especifica la entrada. Los parámetros de entrada se especifican normalmente en el flujo de entrada. Sin embargo, se pueden definir en un conjunto de datos en miembro de un conjunto de datos particionados.

Parámetros simbólicos

Los siguientes parámetros simbólicos se definen en DFHJS2LS:

JAVADIR = *vía de acceso*

Especifica el nombre del directorio Java que utiliza DFHJS2LS. El valor de este parámetro se añade a /usr/lpp/ para crear un nombre de vía de acceso completo de /usr/lpp/ *vía de acceso* .

Generalmente, no es necesario especificar este parámetro. El valor predeterminado es el valor proporcionado para el trabajo de instalación (DFHISTAR) de CICS en el parámetro **JAVADIR**.

PATHPREF = *prefijo*

Especifica un prefijo que amplía la vía de acceso del directorio z/OS UNIX que se utiliza en otros parámetros o '' (serie vacía) si no se utiliza un prefijo.

Generalmente, no es necesario especificar este parámetro. El valor predeterminado es el valor proporcionado para el trabajo de instalación (DFHISTAR) de CICS en el parámetro **PATHPREF**.

SERVICE = *valor*

Utilice este parámetro sólo cuando el soporte de IBM le indique que lo haga.

TMPDIR = *tmpdir*

Especifica la ubicación de un directorio en z/OS UNIX que DFHJS2LS utiliza como espacio de trabajo temporal. El ID de usuario bajo el que se ejecuta el trabajo debe tener permiso de lectura y escritura para este directorio.

El valor predeterminado es /tmp.

TMPFILE = *tmpprefix*

Especifica un prefijo que DFHJS2LS utiliza para construir los nombres de los archivos de espacio de trabajo temporal.

El valor predeterminado es JS2LS.

USSDIR = *vía de acceso*

Especifica el nombre del directorio CICS TS en el sistema de archivos de los servicios del sistema UNIX. El valor de este parámetro se añade a /usr/lpp/cicsts/ para crear un nombre de vía de acceso completo de /usr/lpp/cicsts/ *vía de acceso* . Esto debe especificarse como '.' (punto) si se utiliza el valor predeterminado.

Generalmente, no es necesario especificar este parámetro. El valor predeterminado es el valor proporcionado para el trabajo de instalación (DFHISTAR) de CICS en el parámetro **USSDIR**.

Espacio de trabajo temporal

DFHJS2LS crea estos tres archivos temporales en tiempo de ejecución:

```
tmpdir / tmpprefix .in
tmpdir / tmpprefix .out
tmpdir / tmpprefix .err
```

donde:

tmpdir es el valor que se especifica en el parámetro **TMPDIR**.

tmpprefix es el valor que se especifica en el parámetro **TMPFILE**.

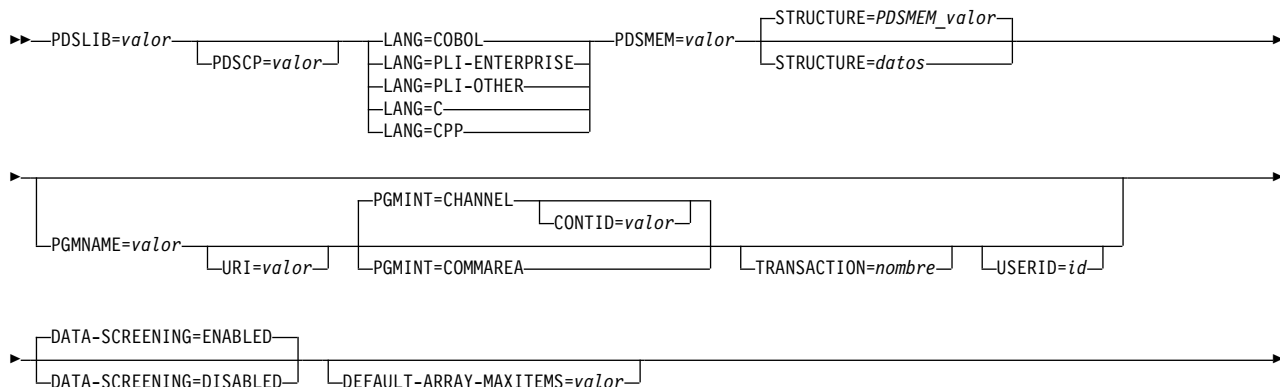
Los nombres predeterminados para los archivos, cuando no se especifica **TMPDIR** y **TMPFILE** son los siguientes:

```
/tmp/JS2LS.in
/tmp/JS2LS.out
/tmp/JS2LS.err
```

Importante: DFHJS2LS no bloquea el acceso a los archivos z/OS UNIX o los miembros del conjunto de datos. Por lo tanto, si dos o más instancias de DFHJS2LS se ejecutan de forma simultánea y utilizan los mismos archivos de espacio de trabajo temporal, nada impide a un trabajo sobrescribir los archivos de espacio de trabajo mientras otro trabajo los está utilizando, llevando a errores imprevisibles.

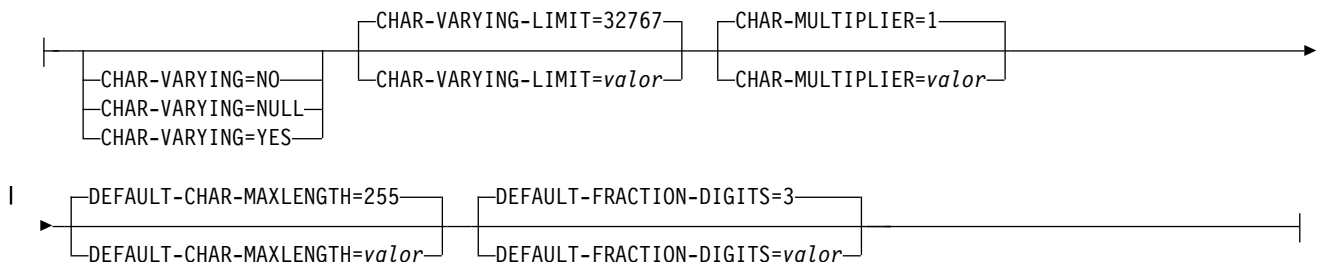
Por lo tanto, se aconseja crear un convenio de denominación y procedimientos operativos que eviten esta situación. Por ejemplo, puede utilizar el parámetro simbólico de sistema **SYSUID** para generar nombres de archivo de espacios de trabajo que son exclusivos para un usuario individual. Estos archivos temporales se suprimen antes de la finalización del trabajo.

Parámetros de entrada de DFHJS2LS

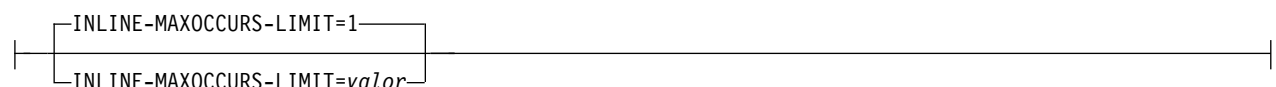




Correlación de datos avanzada (nivel de correlación 1.2 y superior):



Correlación de datos avanzada (nivel de correlación 2.1 y superior):



Utilización de parámetros

- Puede especificar los parámetros de entrada en cualquier orden.
- Cada parámetro se debe iniciar en una nueva línea.
- Un parámetro, y su carácter de continuación, si utiliza uno, no deben sobrepasar la columna 72; las columnas 73 a 80 deben contener espacios en blanco.
- Si un parámetro es demasiado largo para ajustarse en una única línea, utilice el carácter de asterisco (*) al final de la línea para indicar que el parámetro continúa en la línea siguiente. Todo, incluidos los espacios antes del asterisco, se considera parte del parámetro. Por ejemplo:

```
WSBIND=wsbinddir*  
      /app1
```

es equivalente a

```
WSBIND=wsbinddir/app1
```

- Un carácter # en la primera posición de carácter de la línea es un carácter de comentario. La línea se ignora.
- Una coma en la última posición de carácter de la línea es un separador de línea opcional y se ignora.

Descripciones de parámetros

CCSID = *valor*

Especifica el CCSID que se utiliza en tiempo de ejecución para codificar datos de caracteres en la estructura de datos de la aplicación. El valor de este parámetro sustituye el valor del parámetro de inicialización del sistema **LOCALCCSID**. El *valor* debe ser un EBCDIC CCSID soportado por Java y . Si no especifica este parámetro, la estructura de datos de aplicación se codifica utilizando el CCSID especificado en el parámetro de inicialización del sistema.

CHAR-MULTIPLIER = { 1 | *valor* }

Especifica el número de bytes que se permiten para cada carácter cuando el nivel de correlación es 1.2 o posterior. El *valor* de este parámetro puede ser un entero positivo en el rango de 1 a 2.147.483.647. Todas las correlaciones basadas en caracteres no numéricos están sujetas a este multiplicador. Los campos binario, numérico, con zona y decimal empaquetado no están sujetos a este multiplicador.

Este parámetro puede ser útil si, por ejemplo, está planificando utilizar caracteres DBCS donde puede optar por un multiplicador de 3 para permitir espacio para caracteres de cambio y desplazamiento potenciales alrededor de todos los caracteres de doble byte en tiempo de ejecución.

Cuando establece **CCSID=1200** (indicando UTF-16), los únicos valores válidos para **CHAR-MULTIPLIER** son 2 o 4 . Cuando utiliza UTF-16, el valor predeterminado es 2 . Utilice **CHAR-MULTIPLIER=2** cuando espera que los datos de aplicación contengan caracteres que requieren 1 unidad de codificación UTF-16. Utilice **CHAR-MULTIPLIER=4** cuando espera que los datos de aplicación contengan caracteres que requieren 2 unidades de codificación UTF-16.

Nota: Establecer **CHAR-MULTIPLIER** en 1 no excluye el uso de caracteres DBCS, y establecerlo en 2 no excluye el uso de pares de sustitución UTF-16. Sin embargo, si los caracteres amplios utilizan de forma rutinaria algunos valores válidos no se ajustarán al campo asignado. Si se utiliza un valor **CHAR-MULTIPLIER** mayor, es posible almacenar más caracteres en el campo asignado que los que son válidos en el XML. Se debe tener cuidado para ajustarse a las restricciones de rango adecuadas.

CHAR-VARYING = NO | NULL | YES

Especifica cómo se correlacionan datos de caracteres de longitud variable cuando el nivel de correlación es 1.2 o superior. Los tipos de datos binarios de longitud variable se correlacionan siempre con un contenedor o una estructura variable. Si no especifica este parámetro, la correlación predeterminada depende del lenguaje especificado. Puede seleccionar estas opciones:

NO Los datos de caracteres de longitud variable se correlacionan como series de longitud fija.

NULL Los datos de caracteres de longitud variable se correlacionan con series terminadas en nulo.

YES Los datos de caracteres de longitud variable se correlacionan con un tipo de datos CHAR VARYING en PL/I. En los lenguajes COBOL, C y C++, los datos de caracteres de longitud variable se correlacionan con una representación equivalente que consta de dos elementos relacionados: la longitud de datos y los datos.

CHAR-VARYING-LIMIT = 32767 | *valor*

Especifica el tamaño máximo de datos binarios y de datos de caracteres de longitud variable que se correlacionan con la estructura de lenguaje cuando el nivel de correlación es 1.2 o superior. Si los datos binarios o de caracteres son mayores que el valor especificado en este parámetro, se correlacionan con un contenedor y el nombre de contenedor se utiliza en la estructura de lenguaje generada. El valor puede estar comprendido entre 0 y el valor predeterminado 32.767 bytes.

CHAR-WHITESPACE = COLLAPSE | REPLACE | PRESERVE

Especifica cómo CICS manejará el espacio en blanco en los valores de tipo cadena.

COLLAPSE

Los espacios en blanco de inicio, final e incluidos se eliminan y todas las pestañas, líneas nuevas y espacios consecutivos se sustituyen por caracteres de un único espacio.

REPLACE

Cualquier pestaña o línea nueva se sustituye por el número de espacios adecuado.

PRESERVE

Retiene cualquier espacio en blanco en el valor de datos.

Si el parámetro **CHAR-WHITESPACE** no está establecido, el espacio en blanco se contraerá.

Nota: Este parámetro no se aplica a campos con un formato de date-time , uri , base64Binary o hexBinary , donde el espacio en blanco siempre se contrae.

CONTID = *valor*

En un proveedor de servicios, especifique el nombre del contenedor que tiene la estructura de datos de nivel superior utilizada para representar un mensaje JSON.

La longitud del contenedor que CICS pasa al programa de aplicación de destino es mayor que las longitudes del contenedor de solicitud y el contenedor de respuesta.

DATA-SCREENING = { ENABLED | DISABLED }

Especifica si los datos proporcionados por la aplicación se analizan para detectar errores.

ENABLED

Los datos de tiempo de ejecución proporcionados por la aplicación que son incoherentes con la estructura de lenguaje, se tratan como un error y se emite un mensaje DFHPI1010. Se devuelve una respuesta de error a la aplicación.

DISABLED

Los valores de datos de los datos de tiempo de ejecución proporcionados por la aplicación que son incoherentes con la estructura de lenguaje se sustituyen por valores predeterminados. Por ejemplo, un cero sustituye un valor erróneo en un campo numérico. No se emite mensaje DFHPI1010 y se devuelve una respuesta normal a la aplicación. Esta característica se puede utilizar para evitar las respuestas de error INVALID_PACKED_DEC y INVALID_ZONED_DEC generadas desde los campos de salida no inicializados.

DATA-TRUNCATION = { DISABLED | ENABLED }

Especifica si los datos de longitud variable se toleran en una estructura de campo de longitud fija:

DISABLED

Si los datos son menores que la longitud fija que está esperando CICS, CICS rechaza los datos truncados y emite un mensaje de error.

ENABLED

Si los datos son menores que la longitud fija que está esperando CICS, CICS tolera los datos truncados y procesa los datos que faltan como valores nulos

DATETIME = PACKED15 | STRING

Especifica cómo se correlacionan los elementos de fecha y hora JSON con la estructura de lenguaje.

PACKED15

El valor predeterminado es que cualquier elemento de fecha y hora JSON se procesa como una indicación de fecha y hora y se correlaciona con el formato ABSTIME de CICS.

STRING

El elemento de fecha y hora de JSON se procesa como texto.

DEFAULT-ARRAY-MAXITEMS = *valor*

Especifica el límite de matriz máximo a aplicar donde no se implica información de aparición máxima (maxItems) en el esquema JSON. Si no se establece este parámetro, no se aplica límite máximo. El valor de este parámetro puede ser un entero positivo en el rango de 1 a 2147483647. Este parámetro se puede combinar con el uso del parámetro **INLINE-MAXOCCURS-LIMIT** para que influya en la forma en la que se correlacionan las matrices JSON en las estructuras de lenguaje.

DEFAULT-CHAR-MAXLENGTH = 255 | *valor*

Especifica la longitud de matriz predeterminada de los datos de caracteres en caracteres para correlaciones donde no está implicada la longitud en el documento de descripción servicio web, cuando el nivel de correlación es 1.2 o posterior. El valor de este parámetro puede ser un entero positivo en el rango de 1 a 2.147.483.647.

DEFAULT-FRACTION-DIGITS = { 3 | *valor* }

Especifica el número de dígitos de fracción predeterminado a utilizar en un tipo de esquema decimal XML. El valor predeterminado es 3. Para COBOL, el rango válido es 0-17, o 0-30 si se está utilizando **WIDE-COMP3**. Para C o PLI, el rango válido es 0-30.

HTTP-METHODS = { *GET* | *POST* | *PUT* | *DELETE* | *HEAD* }, { *GET* | *POST* | *PUT* | *DELETE* | *HEAD* }, *

Este parámetro es opcional.

El valor predeterminado es para que GET, POST, PUT y DELETE se establezcan, lo que le dice a DFHJS2LS que la aplicación soporta las cuatro operaciones RESTful principales.

Si se proporciona un valor, DFHJS2LS compila un archivo WSBind en el que sólo se aceptan los métodos HTTP especificados explícitamente.

Si una aplicación desea implementar el método HEAD, debe optar deliberadamente por hacerlo. De forma predeterminada, DFHJS2LS supone que la aplicación no soporta los métodos HTTP HEAD de entrada.

Si un cliente JSON intenta utilizar un método HTTP no soportado CICS responde con una respuesta HTTP 405.

INLINE-MAXOCCURS-LIMIT = 1 | *valor*

Especifica si se utiliza el contenido de repetición variable en línea basado en la palabra clave de esquema JSON `maxItems`. El contenido de repetición variable que se correlaciona en línea se coloca en el contenedor actual con el resto de la estructura de lenguaje generada. El contenido de repetición variable se almacena en dos partes, como contador que almacena el número de apariciones de los datos y como una matriz que almacena cada aparición de los datos. La correlación alternativa para el contenido de repetición variable es una correlación basada en contenido, que almacena el número de apariciones de los datos y el nombre del contenedor donde se colocan los datos. Almacenar los datos en un contenedor distinto tiene implicaciones de rendimiento que pueden hacer que sea preferible la correlación en línea.

El parámetro **INLINE-MAXOCCURS-LIMIT** está disponible sólo del nivel de correlación 2.1 en adelante. El valor de **INLINE-MAXOCCURS-LIMIT** puede ser un entero positivo en el rango de 0 a 32.767. Un valor de 0 indica que no se utiliza la correlación en línea. Un valor de 1 garantiza que los elementos opcionales se correlacionan en línea. Si el *valor* del atributo `maxOccurs` es mayor del *valor* de **INLINE-MAXOCCURS-LIMIT**, se utiliza la correlación basada en contenedor; de lo contrario, se utiliza la correlación en línea.

Cuando decide si desea que las listas de repetición variable se correlacionen en línea, tenga en cuenta la longitud de un único elemento de datos recurrentes. Si se producen muchas instancias de longitud larga, es preferible la correlación basada en contenedor; si se producen muchas instancias de longitud corta, es preferible la correlación en línea.

JSON-SCHEMA-RESTFUL = *valor*

Este es un parámetro obligatorio.

El valor indica la ubicación de los servicios del sistema de UNIX donde se almacena el esquema JSON de respuesta.

LANG = **COBOL**

Especifica que el lenguaje de programación de la estructura de lenguaje de alto nivel es COBOL.

LANG = PLI-ENTERPRISE

Especifica que el lenguaje de programación de la estructura de lenguaje de alto nivel es Enterprise PL/I.

LANG = PLI-OTHER

Especifica que el lenguaje de programación de la estructura de lenguaje de alto nivel es un nivel de PL/I distinto a Enterprise PL/I.

LANG = C

Especifica que el lenguaje de programación de la estructura de lenguaje de alto nivel es C.

LANG = CPP

Especifica que el lenguaje de programación de la estructura de lenguaje de alto nivel es C++.

LOGFILE = *valor*

El nombre completo del archivo de z/OS UNIX en el que DFHJS2LS escribe la información de rastreo y registro de actividad. DFHJS2LS crea el archivo, pero no la estructura de directorio, si no existe.

Normalmente, no utiliza este archivo, pero puede solicitarlo la organización de servicio de IBM si encuentra problemas con DFHJS2LS.

MAPPING-LEVEL = { 1.0 | 1.1 | 1.2 | 2.0 | 2.1 | 2.2 | 3.0 | 4.0 | 4.1 }

Especifica el nivel de correlación que utiliza DFHJS2LS cuando genera el archivo de enlace de servicio web y la estructura de lenguaje. Puede seleccionar estas opciones:

- 1.0** Esta opción se retiene para la compatibilidad con servicios web SOAP. No se recomienda utilizar con JSON.
- 1.1** Esta opción se retiene para la compatibilidad con servicios web SOAP. No se recomienda utilizar con JSON.
- 1.2** Esta opción se retiene para la compatibilidad con servicios web SOAP. No se recomienda utilizar con JSON.
- 2.0** Esta opción se retiene para la compatibilidad con servicios web SOAP. No se recomienda utilizar con JSON.
- 2.1** Esta opción se retiene para la compatibilidad con servicios web SOAP. No se recomienda utilizar con JSON.
- 2.2** Esta opción se retiene para la compatibilidad con servicios web SOAP. No se recomienda utilizar con JSON.
- 3.0** Utilice este nivel de correlación para generar un esquema JSON utilizando el conjunto de opciones completo disponible.
- 4.0** Utilice este nivel de correlación con una región de CICS TS 5.2 cuando dese utilizar UTF-16.
- 4.1** Para un soporte de matriz truncable, utilice este nivel de correlación con una región de CICS TS 5.2 que tiene aplicado APAR PI67641, o cualquier región de CICS TS 5.3 o posterior.

Para obtener más información sobre los niveles de correlación, consulte Mapping levels for the CICS JSON assistants.

MAPPING-OVERRIDES = { SAME-AS-MAPPING-LEVEL | HYPHENS-AS-UNDERSCORES | INTEGER-AS-PIC9 | LESS-DUP-NAMES | UNDERSCORES-AS-HYPHENS }

Especifica si el comportamiento predeterminado se sustituye para el nivel de correlación especificado cuando se generan estructuras de lenguaje.

SAME-AS-MAPPING-LEVEL

Este parámetro genera estructuras de lenguaje en el mismo estilo que el nivel de correlación. Es el valor predeterminado.

HYPHENS-AS-UNDERScores

Sólo para PL/I. Este parámetro convierte cualquier guión del esquema JSON en guiones bajos en lugar del carácter X, para mejorar la legibilidad de las estructuras de lenguaje PL/I generadas. Para obtener más información, consulte el apartado JSON schema to PL/I mapping.

INTEGER-AS-PIC9

Únicamente para COBOL y DFHJS2LS. Este parámetro genera estructuras de lenguaje que contienen valores enteros del esquema JSON como numerales en lugar de caracteres alfanuméricos.

LESS-DUP-NAMES

Este parámetro genera nombres de campo de estructura no estructural con `_value` al final del nombre para habilitar la referencia directa al campo. Por ejemplo, en la siguiente estructura PLI, cuando se especifica `MAPPING-OVERRIDES=LESS-DUP-NAMES`, el campo de nivel 12 `streetName` lleva el sufijo `_value`:

```
09 streetName,  
12 streetName CHAR(255) VARYING  
UNALIGNED,  
12 filler BIT (7),  
12 attr_nil_streetName_value BIT (1),
```

La estructura resultante es la siguiente:

```
09 streetName,  
12 streetName_value CHAR(255) VARYING  
UNALIGNED,  
12 filler BIT (7),  
12 attr_nil_streetName_value BIT (1),
```

UNDERScores-AS-HYPHENS

Esta opción está habilitada automáticamente en el nivel de correlación 4.0.

Solo para COBOL. Este parámetro convierte cualquier guión bajo en esquema JSON en guiones, en lugar del carácter X, para mejorar la legibilidad de las estructuras de lenguaje COBOL generadas. Si se produce un conflicto de nombres de campo, los campos se numeran para garantizar que con exclusivos. Para obtener más información, consulte el apartado JSON schema to COBOL mapping.

MINIMUM-RUNTIME-LEVEL = { **MINIMUM** | **1.0** | **1.1** | **1.2** | **2.0** | **2.1** | **2.2** | **3.0** | **4.0** | **4.1** | **CURRENT** }

Especifica el entorno de ejecución de CICS mínimo en el que se puede desplegar el archivo de enlace de servicio web. Si selecciona un nivel que no coincide con los demás parámetros que ha especificado, recibe un mensaje de error. Puede seleccionar estas opciones:

MINIMUM

El nivel de tiempo de ejecución más bajo posible de CICS se asigna automáticamente a los parámetros determinados que seleccione.

- 1.0** Esta opción se retiene para la compatibilidad con servicios web SOAP. No se recomienda utilizar con JSON.
- 1.1** Esta opción se retiene para la compatibilidad con servicios web SOAP. No se recomienda utilizar con JSON.

- 1.2 Esta opción se retiene para la compatibilidad con servicios web SOAP. No se recomienda utilizar con JSON.
- 2.0 Esta opción se retiene para la compatibilidad con servicios web SOAP. No se recomienda utilizar con JSON.
- 2.1 Esta opción se retiene para la compatibilidad con servicios web SOAP. No se recomienda utilizar con JSON.
- 2.2 Esta opción se retiene para la compatibilidad con servicios web SOAP. No se recomienda utilizar con JSON.
- 3.0 El archivo de enlace de servicios web generado se despliega en una región de CICS TS 4.1 o posterior.

Nota: El soporte JSON sólo está disponible desde CICS TS 4.2 en adelante.

- 4.0 El archivo de enlace de servicios web generado se despliega correctamente en una región de CICS TS 5.2 o posterior. Con este nivel de ejecución, puede utilizar un nivel de correlación de 4.0 o anterior para el parámetro **MAPPING-LEVEL**. Puede utilizar cualquier parámetro opcional en este nivel.
- 4.1 El archivo de enlace de servicios web generado se despliega correctamente en una región de CICS TS V5.2 que tiene aplicado APAR PI67641, o en cualquier CICS TS V5.3 o región posterior. Con este nivel de ejecución, puede utilizar un nivel de correlación de 4.1 o anterior para el parámetro **MAPPING-LEVEL**.

CURRENT

El archivo de enlace de servicios web generado se despliega correctamente en una región de CICS al mismo nivel de ejecución que el que está utilizando para generar el archivo de enlace de servicio web.

NAME-TRUNCATION = { **LEFT** | **RIGHT** }

Especifica si los nombres JSON están trucados desde la izquierda o la derecha. El asistente de servicios web de CICS trunca nombres JSON en la longitud adecuada para el lenguaje de alto nivel especificado; de forma predeterminada los nombres se truncan desde la derecha.

PDSCP = *valor*

Especifica la página de códigos que se utiliza en los miembros del conjunto de datos particionados que se especifican en el parámetro **PDSMEM**, donde el *valor* es un número CCSID o un número de página de códigos Java. Si no se especifica este parámetro, se utiliza la página de códigos de los servicios del sistema z/OS UNIX. Por ejemplo, puede especificar **PDSCP** = 037.

PDSLIB = *valor*

Especifica el nombre del conjunto de datos particionados que contiene el lenguaje de alto nivel generado. Los miembros del conjunto de datos que se utilizan para la solicitud y respuesta se especifican en el parámetro **PDSMEM**.

PDSMEM = *valor*

Especifica el prefijo de 1 a 6 caracteres que utiliza DFHJS2LS para generar el nombre del miembro del conjunto de datos particionados que contendrá las estructuras de lenguaje de alto nivel.

DFHJS2LS genera un miembro de conjunto de datos particionados añadiendo 01 al prefijo.

PGMINT = CHANNEL | COMMAREA

Para un proveedor de servicios, especifique cómo CICS pasa datos al programa de aplicación de destino:

CHANNEL

CICS utiliza una interfaz de canal para pasar datos al programa de aplicación de destino.

COMMAREA

CICS utiliza un área de comunicación para pasar datos al programa de aplicación de destino.

Cuando el programa de aplicación de destino ha procesado la solicitud, debe utilizar el mismo mecanismo para devolver la respuesta. Si se ha recibido la solicitud en un área de comunicación, la respuesta debe devolverse en el área de comunicación; si la solicitud se ha recibido en un contenedor, la respuesta debe devolverse en un contenedor. La longitud del contenedor o área de comunicaciones que pasa CICS al programa de aplicación de destino es mayor que las longitudes del contenedor o área de comunicación de solicitud y el contenedor o área de comunicación de la respuesta.

PGMNAME = *valor*

Especifica el nombre de un recurso PROGRAM de CICS.

Cuando se utiliza DFHJS2LS para generar un archivo de enlace de servicio web que se utiliza en un proveedor de servicios, debe proporcionar este parámetro. Especifica el nombre de recurso del programa de aplicación que se expone como un servicio web.

Cuando se utiliza DFHJS2LS para generar un archivo de enlace de servicio web que se utiliza en un solicitante de servicio, omita este parámetro.

STRUCTURE = *nombre*

Sólo para C y C++, especifica cómo se genera el nombre de la estructura.

A la estructura generada se le da el nombre de *nombre 01*.

Si el nombre se omite, la estructura tiene el mismo nombre que el nombre del miembro de conjunto de datos particionados generado desde el parámetro **PDSMEM** que especifica.

SYNCONRETURN = { NO | YES }

Especifica si el servicio web remoto puede emitir un punto de sincronización.

NO El servicio web remoto no puede emitir un punto de sincronización. Este valor es el predeterminado. Si el servicio web remoto emite un punto de sincronización, falla con una terminación anómala ADPL.

YES El servicio web remoto puede emitir un punto de sincronización. Si selecciona YES, la tarea remota se confirma como una unidad de trabajo individual cuando se devuelve el control desde el servicio web remoto. Si el servicio web remoto actualiza un recurso recuperable y se produce un fallo después de que se devuelve, la actualización para ese recurso no se puede restituir.

TRANSACTION = *nombre*

En un proveedor de servicios, este parámetro especifica de uno a cuatro caracteres del nombre de una transacción de alias que puede iniciar la interconexión. El valor de este parámetro se utiliza para definir el atributo TRANSACTION del recurso URIMAP cuando se crea automáticamente utilizando el mandato de exploración **PIPELINE**.

Caracteres aceptables:

A-Z a-z 0-9 \$ @ # _ < >

URI = *valor*

En un proveedor de servicios, este parámetro especifica el URI relativo que utiliza un cliente para acceder al servicio web. CICS utiliza el valor que se especifica cuando genera un recurso URIMAP desde un archivo de enlace de servicio web que ha creado DFHJS2LS. El parámetro especifica el componente de vía de acceso del URI al que se aplica la definición URIMAP. Al utilizar comodines * al final de un URI, el valor URI debe ir seguido de coma.

USERID = *id*

En un proveedor de servicios, este parámetro especifica de uno a ocho caracteres del ID de usuario, que puede utilizar cualquier cliente web. Para una respuesta generada por la aplicación o un servicio web, la transacción de alias se conecta bajo este ID de usuario. El valor de este parámetro se utiliza para definir el atributo USERID del recurso URIMAP cuando se crea automáticamente utilizando el mandato de exploración **PIPELINE**.

Caracteres aceptables:

A-Z a-z 0-9 \$ @ #

WIDE-COMP3 = { FULL | NO | YES }

Controla el tamaño máximo de la longitud variable decimal empaquetada en la estructura de lenguaje COBOL o PL/I generada.

FULL Para COBOL y PL/I. DFHJS2LS genera un campo decimal empaquetado que es lo suficientemente grande como para contener todos los valores válidos. El tamaño máximo es de 31 dígitos. Es el valor predeterminado.

NO Solo para COBOL. DFHJS2LS limita la longitud variable de decimal empaquetado a 10 al generar el tipo de estructura de lenguaje COBOL COMP-3. Si el tamaño del decimal empaquetado es mayor de 18, se emite un mensaje DFHPI9022W para indicar que el tipo especificado se está restringiendo a un total de 18 dígitos.

YES Solo para COBOL. DFHJS2LS soporta el tamaño máximo de 31 al generar el tipo de estructura de lenguaje COBOL COMP-3.

Nota: Las opciones NO y YES generan campos que no pueden representar todos los valores válidos; la opción FULL evita este problema. Sin embargo, la opción FULL no permite que algunos valores no válidos se representen en el campo de decimal empaquetado. Por ejemplo, si un esquema indica que hay un máximo de cinco dígitos y un máximo de dos dígitos fraccionales, la opción FULL generará un campo de decimal empaquetado que permite siete dígitos, y esto permite espacio para valores válidos como 25000 y 999,99, pero también proporciona espacio para algunos valores no válidos como 9999,99. Cuando utilice la opción FULL, tenga cuidado de no generar valores no válidos en datos de aplicación.

WSBIND = *valor*

El nombre completo del archivo de enlace de servicio web de z/OS UNIX. DFHJS2LS crea el archivo, pero no la estructura de directorio, si no existe. La extensión de archivo predeterminada es .wsbind.

Otra información

- El ID de usuario bajo el que se ejecuta DFHJS2LS debe configurarse para utilizar los servicios de sistema UNIX. El ID de usuario debe tener permiso de lectura para la estructura de archivo de CICS z/OS UNIX y las bibliotecas PDS y permiso de escritura para los directorio especificados en los parámetros **LOGFILE**, **WSBIND** y **WSDL**.
- El ID de usuario debe tener una asignación de almacenamiento lo suficientemente grande para ejecutar Java.
- El JCL tiene una longitud de parámetro máxima de 100 caracteres. Esto puede aumentarse utilizando la sentencia **STDPARM**. Para obtener más información, consulte el apartado *z/OS UNIX System Services User Guide*.

Ejemplo

```
//JS2LS JOB '  
información de contabilidad  
'  
,  
name,MSGCLASS=A  
// SET QT='''  
//JAVAPROG EXEC DFHJS2LS,  
// TMPFILE=&QT.&SYSUID.&QT  
/INPUT.SYSUT1 DD *  
PDSLIB=//CICSHLQ.SDFHSAMP  
PDSMEM=CPYBK2  
JSON-SCHEMA-RESTFUL=example.json  
LANG=COBOL  
LOGFILE=/u/exampleapp/wsbinding/example.log  
MAPPING-LEVEL=4.0  
CHAR-VARYING=NULL  
INLINE-MAXOCCURS-LIMIT=2  
PGMNAME=DFH0XCMN  
URI=exampleApp/example/*,  
PGMINT=COMMAREA  
SYNCONRETURN=YES  
WSBIND=/u/exampleapp/wsbinding/example.wsbinding  
/*
```

Creación de una aplicación de proveedor de servicios JSON utilizando un asistente de JSON

Puede crear una aplicación de proveedor de servicios desde un esquema JSON que cumple con el esquema JSON v4 (borrador), o desde una estructura de datos de lenguaje de alto nivel. El asistente de CICS JSON le ayuda a desplegar las aplicaciones CICS en una configuración del proveedor de servicios.

Acerca de esta tarea

Cuando utiliza el asistente para desplegar una aplicación CICS como proveedor de servicios, tiene dos opciones:

- Empezar con un esquema JSON y utilizar el asistente para generar estructuras de datos de lenguaje.
Utilice esta opción cuando desee implementar un proveedor de servicios que se ajuste a una descripción de servicio web existente.
- Empiece con las estructuras de datos de lenguaje y utilice el asistente para generar el esquema JSON.
Utilice esta opción cuando desee exponer un programa existente como servicio JSON.

Creación de una aplicación de un proveedor de servicios a partir de un esquema JSON

Utilizando el asistente de JSON de CICS, puede crear una aplicación de proveedor de servicios desde un esquema JSON.

Antes de empezar

Antes de poder crear una aplicación de proveedor de servicio, se deben cumplir las siguientes condiciones:

- La descripción de servicios web debe estar en un archivo UNIX en z/OS y debe crear una interconexión de modalidad de proveedor adecuada en la región de CICS.
- Debe definir como OMVS el ID de usuario con el que se ejecuta DFHJS2LS.
- El ID de usuario debe tener permiso de lectura para las bibliotecas z/OS UNIX y PDS y permiso de escritura para los directorios especificados en los parámetros **LOGFILE** , **WSBIND** y **JSON-SCHEMA-REQUEST** , **JSON-SCHEMA-RESPONSE** y **JSON-SCHEMA-RESTFUL**.
- Debe asignar almacenamiento suficiente para el ID de usuario para que el ID ejecute Java. Puede utilizar cualquier versión soportada de Java. De forma predeterminada, DFHJS2LS utiliza la versión de Java especificada en el parámetro **JAVADIR**.

Acerca de esta tarea

Puede utilizar el asistente de JSON para crear estructuras de lenguaje desde el esquema JSON para la aplicación de proveedor de servicios.

Procedimiento

1. Utilice el programa por lotes DFHJS2LS para generar un archivo de enlace de servicio web y una o varias estructuras de datos de lenguaje. DFHJS2LS contiene un conjunto de parámetros opcionales grande que le proporciona flexibilidad para crear el archivo de enlace y las estructuras de lenguaje que requiere su aplicación. Tenga en cuenta estas opciones cuando habilite una aplicación existente para los servicios web:
 - ¿Qué mecanismos utilizará CICS para pasar datos al programa de aplicación de proveedor de servicios? Puede utilizar canales y pasar datos a contenedores o utilizar un COMMAREA. Se recomiendan los canales y los contenedores. Especifíquelos con el parámetro **PGMINT**.
 - ¿Qué lenguaje desea generar? DFHJS2LS puede generar estructuras de datos de lenguaje COBOL, C/C++ o PL/I. Especifique el lenguaje utilizando el parámetro **LANG**.
 - ¿Qué nivel de correlación desea utilizar? Cuánto mayor sea el nivel de correlación, mayor el control y soporte que tiene disponible para el manejo de caracteres y datos binarios en tiempo de ejecución. Algunos parámetros opcionales sólo están disponibles en niveles de correlación superiores. Se recomienda utilizar el nivel de correlación más alto disponible. Especifique el nivel de correlación con el parámetro **MAPPING-LEVEL**.
 - ¿Qué URI desea que utilice el solicitante de servicio web? Especifique un URI relativo utilizando el parámetro **URI**; por ejemplo, **URI=/my/test/webservice**. CICS utiliza el valor cuando crea el recurso URIMAP.
 - ¿En qué transacción e ID de usuario ejecutará la solicitud y respuesta del servicio web? Puede utilizar una transacción de alias para ejecutar la aplicación para componer una respuesta para el solicitante de servicio. La

transacción de alias se conecta con el ID de usuario. Especifíquelo con los parámetros **TRANSACTION** y **USERID**. Estos valores se utilizan cuando se crea el recurso URIMAP. Si no desea utilizar una transacción específica, no utilice estos parámetros.

Cuando envía DFHJS2LS, CICS genera el archivo de enlace de servicio web y lo coloca en la ubicación que especifica con el parámetro **WSBIND**. Las estructuras de lenguaje se colocan en el conjunto de datos particionados que especifica con el parámetro **PDSLIB**.

2. Copie el archivo de enlace de servicio web generado en el directorio pickup del recurso PIPELINE de modalidad de proveedor que desea utilizar para la aplicación de servicio web. Debe copiar el archivo de enlace en modalidad binaria.
3. Escriba un programa de aplicación de proveedor de servicios para la interfaz con las estructuras de lenguaje generadas e implemente la lógica empresarial necesaria.
4. Utilice el mandato **PIPELINE SCAN** para crear dinámicamente el recurso WEBSERVICE y un recurso URIMAP.
 - El recurso WEBSERVICE encapsula el archivo de enlace de servicio web en CICS y se utiliza en tiempo de ejecución.
 - El recurso URIMAP proporciona a CICS la información necesaria para asociar el recurso WEBSERVICE con un URI concreto.

Como alternativa, puede definir los recursos, aunque no es lo recomendable.

Resultados

Si tiene problemas a la hora de enviar DFHJS2LS, o los recursos no se instalan correctamente, consulte Troubleshooting the JSON assistant.

Creación de una aplicación de un proveedor de servicios a partir de una estructura de datos

Uso del asistente de servicios web de CICS para crear una aplicación de proveedor de servicios de la estructura de datos de lenguaje de nivel superior.

Antes de empezar

Antes de crear una aplicación de proveedor de servicios, asegúrese de que su configuración cumple con estas condiciones previas:

- Las estructuras de datos de lenguaje de alto nivel deben cumplir los siguientes criterios:
 - Las estructuras de datos deben definirse por separado desde el programa de origen; por ejemplo, en un libro de copias COBOL.
 - Si el programa de aplicación PL/I o COBOL utiliza diferentes estructuras de datos para la entrada y la salida, las estructuras de datos deben estar definidas en dos miembros diferentes en un conjunto de datos particionados. Si se utiliza la misma estructura para la entrada y salida, la estructura debe estar definida en un único miembro.

Para C y C++, las estructuras de datos pueden estar en el mismo miembro en un conjunto de datos particionados.
- Las estructuras de lenguaje deben estar disponibles en un conjunto de datos particionados y debe crear un recurso PIPELINE adecuado en la región de CICS.
- Debe definir como OMVS el ID de usuario que DFHLS2JS utiliza para ejecutar.

- El ID de usuario debe tener permiso de lectura para las bibliotecas z/OS UNIX y PDS y permiso de escritura para los directorios especificados en los parámetros de salida **LOGFILE** , **WSBIND** y **JSON-SCHEMA-REQUEST** y **JSON-SCHEMA-RESPONSE**.
- El ID de usuario debe tener una asignación de almacenamiento lo suficientemente grande para ejecutar Java. Puede utilizar cualquier versión soportada de Java. De forma predeterminada, DFHLS2JS utiliza la versión de Java especificada en el parámetro **JAVADIR**.

Procedimiento

Siga estos pasos para crear una aplicación de proveedor de servicios desde una estructura de datos de alto nivel:

1. Si la interfaz de la aplicación de proveedor de servicios utiliza canales y muchos contenedores, cree un documento de descripción de canal que describe la interfaz en JSON. Debe guardar el documento de descripción de canal en un directorio adecuado en z/OS UNIX. CICS utiliza este documento para construir y deconstruir un mensaje JSON desde los contenedores en un canal. De lo contrario, puede utilizar un contenedor en un canal y no crear un documento de descripción de canal.

Para obtener más información sobre cómo crear un documento de descripción de canal, consulte “Creación de un documento de descripción de canal” en la página 277.

2. Utilice el programa por lotes DFHLS2JS para generar un archivo de enlace de servicio web y una descripción de servicio web desde la estructura de lenguaje. El programa por lotes DFHLS2JS se puede encontrar en *HLQ* .XDFHINST donde *HLQ* es la ubicación donde ha instalado CICS. DFHLS2JS contiene un conjunto de parámetros opcionales grande que le proporciona flexibilidad para crear el archivo de enlace y las estructuras de lenguaje que requiere su aplicación. Tenga en cuenta las siguientes opciones cuando habilite los servicios web para una aplicación existente:

- ¿Qué mecanismos desea utilizar CICS para pasar datos al programa de aplicación de proveedor de servicios? Puede utilizar canales y pasar datos a contenedores o utilizar un COMMAREA. Especifique el mecanismo utilizando el parámetro **PGMINT**. Si la interfaz de aplicación utiliza canales y muchos contenedores, especifique el parámetro **REQUEST-CHANNEL** y, opcionalmente, **RESPONSE-CHANNEL**. Puede utilizar estos parámetros sólo cuando el nivel de correlación es 3.0 o superior.
- ¿Qué nivel de correlación desea utilizar? Cuánto mayor sea el nivel de correlación, mayor el control y soporte que tiene disponible para el manejo de caracteres y datos binarios en tiempo de ejecución. Algunos parámetros opcionales sólo están disponibles en niveles de correlación superiores. Debe especificar el nivel de correlación más alto disponible en el parámetro **MAPPING-LEVEL**.
- ¿Qué URI desea que utilice el servicio web? Especifique un URI absoluto utilizando el parámetro **URI**; por ejemplo, **URI** = `http://www.example.org:80/my/test/webservice`. La parte relativa de esta dirección, `/my/test/webservice`, se utiliza al crear el recurso URIMAP.

Cuando envía DFHLS2JS, CICS genera un archivo de enlace de servicio web y lo coloca en la ubicación que ha especificado con el parámetro **WSBIND**. Los esquemas JSON generados se colocan en la ubicación que ha especificado con los parámetros **JSON-SCHEMA-REQUEST** y **JSON-SCHEMA-RESPONSE**.

3. Revise el esquema JSON generado.

Estos esquemas se utilizan para definir los formatos de datos de entrada y salida para el servicio web JSON. El desarrollador de aplicaciones debe utilizar estos esquemas al crear una aplicación para interactuar con el servicio web JSON.

Nota: El cambio de esquema generado invalida el archivo de enlace de servicio web generado, WSBind.

Si desea cambiar el esquema, por ejemplo, para cambiar el nombre de los campos dentro del esquema, debe utilizar DFHJS2LS para generar un archivo de enlace de servicio web nuevo y un nuevo conjunto de estructuras de lenguaje. El programa de aplicación en CICS debe modificarse para utilizar las nuevas estructuras de lenguaje.

4. Copie el archivo de enlace de servicio en el directorio de recogida de la interconexión de modalidad de proveedor que desea utilizar para la aplicación de servicio web. Debe copiar el archivo de enlace de servicio web en modalidad binaria.
5. Use el mandato **PIPELINE SCAN** para crear dinámicamente el recurso WEBSERVICE y un recurso URIMAP.
 - El recurso WEBSERVICE contiene el archivo de enlace de servicio web en CICS y se utiliza en tiempo de ejecución.
 - El recurso URIMAP proporciona a CICS la información necesaria para asociar el recurso WEBSERVICE con un URI concreto.

Como alternativa puede definir los recursos usted mismo.

Resultados

La creación de la aplicación de proveedor de servicios se ha completado.

Si tiene problemas a la hora de enviar DFHLS2JS, o los recursos no se instalan correctamente, consulte Troubleshooting the JSON assistant.

Qué hacer a continuación

Ponga la descripción de servicios web a disposición de todo el que desarrolle un servicio web que accederá al servicio.

Creación de un documento de descripción de canal

Cree un documento de descripción de canal cuando la aplicación de proveedor de servicios utiliza una interfaz de canal con muchos contenedores.

Acerca de esta tarea

Utilice un editor XML para crear un documento de descripción de canal. El esquema para la descripción de canal se denomina `channel.xsd` y está en el directorio `/usr/lpp/cicsts/cicsts54/schemas/channel` (donde `/usr/lpp/cicsts/cicsts54` es el directorio de instalación predeterminado para archivos CICS en z/OS UNIX).

Procedimiento

1. Cree un documento XML con un elemento `<channel>` y el espacio de nombres de canal de CICS:

```
<channel name="myChannel"
xmlns="http://www.ibm.com/xmlns/prod/CICS/channel">
</channel>
```

2. Añada un elemento <container> a cada contenedor que utiliza el programa de aplicación en el canal. Debe utilizar los atributos de nombre, tipo y uso para describir cada contenedor. El ejemplo siguiente muestra seis contenedores con diferentes valores de atributo:

```
<container name="cont1" type="char" use="required"/>
<container name="cont2" type="char" use="optional"/>
<container name="cont3" type="bit" use="required"/>
<container name="cont4" type="bit" use="optional"/>
<container name="cont5" type="bit" use="required">
<structure location="//HLQ.PDSNAME(MEMBER)"/>
</container>
<container name="cont6" type="bit" use="optional">
<structure location="//HLQ.PDSNAME(MEMBER2)"/>
</container>
```

El elemento de estructura indica que el contenido se define en una estructura de lenguaje ubicada en un miembro de conjunto de datos particionados.

3. Guarde el documento XML en z/OS UNIX.

Esquema de canal

El documento de descripción de canal debe ajustarse al siguiente esquema:

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.ibm.com/xmlns/prod/CICS/channel"
xmlns:tns="http://www.ibm.com/xmlns/prod/CICS/channel"
elementFormDefault="qualified">
<element name="channel">
1
<complexType>
<sequence>
<element name="container" maxOccurs="unbounded" "unbounded" minOccurs="0">
2
<complexType>
<sequence>
<element name="structure" minOccurs="0">
3
<complexType>
<attribute name="location" type="string" use="required"/>
<attribute name="structure" type="string" use="optional"/>
</complexType>
</element>
</sequence>
<attribute name="name" type="tns:name16Type" use="required"/>
<attribute name="type" type="tns:typeType" use="required"/>
<attribute name="use" type="tns:useType" use="required"/>
</complexType>
</element>
</sequence>
<attribute name="name" type="tns:name16Type" use="optional" />
</complexType>
</element>
<simpleType name="name16Type">
<restriction base="string">
<maxLength value="16"/>
</restriction>
</simpleType>
<simpleType name="typeType">
<restriction base="string">
<enumeration value="char"/>
<enumeration value="bit"/>
</restriction>
</simpleType>
<simpleType name="useType">
<restriction base="string">
<enumeration value="required"/>
```

```

<enumeration value="optional"/>
</restriction>
</simpleType>
</schema>

```

1. Este elemento representa un canal de CICS.
2. Este elemento representa un contenedor de CICS dentro del canal.
3. Una estructura solo se puede utilizar con contenedores en modo 'bits'. El atributo 'ubicación' indica la ubicación de un archivo que correlaciona el contenido del contenedor. El atributo 'estructura' puede utilizarse en C y C++ para indicar el nombre de la estructura.

Qué hacer a continuación

Ejecute DFHLS2JS para crear correlaciones y un esquema JSON para la aplicación de proveedor de servicios web. DFHLS2JS sitúa las correlaciones para el canal en el esquema JSON en el orden en el que se especifican los contenedores en el documento de descripción de canal.

Personalización de esquemas JSON generados

Los esquemas JSON generados por DFHLS2JS tienen contenido generado automáticamente que puede ser conveniente que lo cambie antes de su publicación. La personalización de esquemas JSON puede dar como resultado volver a generar el archivo de enlace de servicio web y, en algunos casos, escribir un programa derivador.

Acerca de esta tarea

Siga estos pasos para personalizar los esquemas JSON generados:

Procedimiento

1. Para hacer publicidad del soporte para HTTPS, utilice el parámetro **URI** en DFHLS2JS para establecer un URI absoluto.
2. Para proporcionar la ubicación de red del servicio web, utilice el parámetro **URI** en DFHLS2JS para establecer un URI absoluto.
3. Piense en si los nombres generados automáticamente en el esquema JSON son adecuados para su fines. Puede cambiar el nombre de las propiedades.
Estos valores forman parte de la interfaz de programación a la que codifica un programa cliente. Si los nombres generados no son lo suficientemente significativos, el mantenimiento del código de aplicación puede ser más difícil durante un largo periodo de tiempo.
Si cambia cualquiera de estos valores, debe utilizar DFHJS2LS para volver a generar el archivo de enlace de servicios web. Las estructuras de lenguaje que se crean es probable que no sean compatibles con la aplicación existente, por lo tanto se pueden necesitar cambios de aplicación. Revise las estructuras de lenguaje generadas, y piense en escribir un programa de derivador como se indicó en el paso 4.
4. Piense si los campos COMMAREA expuestos en los esquemas JSON son adecuados. Puede pensar en eliminar los datos que no son útiles para el desarrollador de clientes JSON:
 - Los campos que se utilizan únicamente para valores de salida se pueden eliminar del esquema que se correlaciona con las estructuras de datos de entrada.
 - Campos de relleno.
 - Anotaciones generadas de forma automática.

Si realiza cualquiera de estos cambios, debe volver a generar el archivo de enlace de servicios web utilizando DFHJS2LS. Las nuevas estructuras de lenguaje generadas no son compatibles con las estructuras de lenguaje originales, por lo que debe escribir un programa derivador para correlacionar datos desde la nueva representación a la antigua. Este programa derivador debe ejecutar un mandato **EXEC CICS LINK** para el programa de aplicación de destino y, después, correlacionar los datos devueltos.

Este nivel de personalización requiere el máximo esfuerzo, pero da como resultado las interfaces de programación más significativas para los desarrolladores de clientes JSON.

Resultados

Tiene un esquema JSON personalizado que coincide con los requisitos empresariales y un PROGRAM en CICS que los implementa.

Creación de una aplicación de proveedor de servicios web RESTful

La implementación de CICS de los servicios web de JSON RESTful es similar a la de los servicios web SOAP. La mayoría de los conceptos y arquitectura son compartidos, pero CICS requiere el uso de un esquema JSON.

Acerca de esta tarea

La implementación de un servicio web de RESTful JSON implica las siguientes tareas:

Procedimiento

1. Generar la interfaz de aplicación.

Entrada:

- El esquema JSON que define el modelo de datos para el servicio web RESTful.

Salida:

- Las estructuras de lenguaje (por ejemplo, el libro de copias COBOL) que correlaciona el esquema JSON.
- Un archivo WSBind.

Ejecute el programa de utilidad DFHJS2LS y especifique los parámetros de entrada adecuados. Estos parámetros incluyen:

- La ubicación del esquema JSON.
- La lista de métodos soportados (GET, PUT, POST y DELETE están habilitados de forma predeterminada).
- El URI en el que se despliega el servicio.
- El nombre del PROGRAM de aplicación que implementa el servicio.
- Todos los parámetros de correlación de datos que sean necesarios.

2. Cree una aplicación que utilice esta interfaz.

Entrada:

- Las estructuras de lenguaje desde el paso 1.
- Un conocimiento de qué operaciones RESTful se implementarán

Salida:

- Un programa adecuado para el despliegue en CICS

Escriba un programa que realice lo siguiente:

Nota: Si proporciona el nombre de su propio contenedor en el parámetro **CONTID** para DFHJS2LS, debe utilizar este contenedor en lugar de DFHWS-DATA siempre que se mencione en los pasos siguientes.

- a. Examine el URI para entender la identidad del recurso. CICS proporciona varios contenedores para ayudarle a identificar componentes interesantes del URI. Los contenedores se describen en Tabla 15 . Los ejemplos muestran el contenido de cada contenedor para el URI:

`http://www.example.org:10000/JSONServices/CustomerDetails/13388?action=query`

Si el URIMAP que coincide tiene la PATH de /JSONServices/
CustomerDetails/*

Tabla 15. Contenedores DFHWS-URI. Contenedores DFHWS-URI

Contenedor	Contenido	Ejemplo
DFHWS-URI	El URI completo	<code>http:// www.example.org:10000/ JSONServices/ CustomerDetails/ 13388?action=query</code>
	La parte de la ruta de URI que coincide con URIMAP	<code>/JSONServices/ CustomerDetails/*</code>
DFHWS-URI-RESID	La ruta de URI con la parte que coincide con la URIMAP eliminada (<i>el identificador de recurso</i>)	<code>13388</code>
DFHWS-URI-QUERY	Serie de consulta	<code>action=query</code>

- b. Validar el URI. Si se produce un problema, notifíquelo a CICS y finalice.
- c. Consulte el contenedor DFHHTTPMETHOD para determinar qué método se está llevando a cabo. Para obtener más información, consulte el apartado DFHHTTPMETHOD.
- d. Para un método POST (crear) (si es necesario):
 - Lea los datos de entrada del contenedor DFHWS-DATA.
 - Interprete los datos utilizando las estructuras de lenguaje generadas en el paso 1 en la página 280.
 - Valide los datos. Si se produce un problema, notifíquelo a CICS y finalice.
 - Lleve a cabo el proceso específico de la aplicación necesario para crear el *recurso*.
 - Opcionalmente, escriba en el contenedor DFHRESPONSE para notificar al cliente del identificador del nuevo recurso. CICS no transforma el contenido de este contenedor, pero lo envía directamente en la respuesta HTTP. El contenedor DFHWS-DATA se ignora.
- e. Si se necesita un método GET (consultar) o HEAD (consultar), escriba los datos que representan el *recurso* en el contenedor DFHWS-DATA.
- f. Si se necesita un método PUT (establecer):
 - Lea los datos de entrada del contenedor DFHWS-DATA.
 - Interprete los datos utilizando las estructuras de lenguaje generadas en el paso 1 en la página 280.
 - Valide los datos. Si se produce un problema, notifíquelo a CICS y finalice.
 - Lleve a cabo el proceso específico de la aplicación necesario para actualizar el *recurso*.

- g. Si se necesita un método DELETE, lleve a cabo cualquier proceso específico de la aplicación necesario para suprimir el *recurso*.

Nota: El modelo de datos RESTful implementado por CICS no envía un cuerpo de respuesta para los métodos PUT, POST o DELETE de forma predeterminada. Las aplicaciones RESTful normalmente utilizan el código de estado HTTP para indicar éxito o fracaso. Si la aplicación se completa normalmente, CICS envía una respuesta de 200 (OK). Para obtener más información sobre el envío de respuestas de error, consulte Application error reporting. Si desea enviar un cuerpo de respuesta para un método PUT, POST o DELETE, debe escribir el contenedor DFHRESPONSE. Si está presente, CICS envía el contenido de este contenedor del cuerpo HTTP sin más proceso. CICS ignora el contenedor DFHWS-DATA en el proceso de respuesta para estos métodos.

3. Desplegar los artefactos.

Entrada:

- El archivo WSBind desde el paso 1 en la página 280.
- El programa CICS desde el paso 2 en la página 280.
- Un recurso de interconexión de modalidad de proveedor de CICS configurado para JSON.

Salida:

- Un servicio RESTful JSON desplegado.

Despliegue el programa para CICS de forma normal.

Bien:

- Desplegando el archivo WSBind en el directorio de recogida de la interconexión y, a continuación, emitiendo un mandato **PIPELINE SCAN** para crear los recursos WEBSERVICE y URIMAP.
- O definiendo e instalando manualmente un recurso WEBSERVICE y el recurso URIMAP asociado. El URIMAP debe asociar el URI a PIPELINE y a WEBSERVICE.

4. Pruebe el servicio.

Entrada:

- El esquema JSON.
- El URI del servicio web RESTful.
- El servicio desplegado desde el paso 3.
- El cliente de prueba de JSON que elija.

Salida:

- Una solicitud gestionada correctamente.

Utilice el cliente de prueba de su elección para enviar una solicitud JSON a CICS.

Si recibe una respuesta inesperada, intente la determinación de problemas. Para obtener más información, consulte el apartado Troubleshooting problems with JSON requests.

Consideraciones de diseño para aplicaciones de proveedor de servicios web RESTful

Este tema describe algunos problemas que debe tener en cuenta cuando planifica y diseña una aplicación de proveedor de servicios web RESTful para JSON.

Recopilaciones de recursos

Un diseño habitual para las API de RESTful es soportar la recuperación de recopilaciones de recursos. Por ejemplo, es posible que exista un servicio que devuelva un conjunto de objetos de la siguiente manera:

```
GET /Services/CustomerDetails?Surname=Cooper
```

Se espera que esta solicitud devuelva información sobre todos los objetos CustomerDetails donde el apellido es "Cooper". Los objetos CustomerDetails individuales pueden devolverse utilizando una URI más específica como:

```
GET /Services/CustomerDetails/Customer27
```

En este ejemplo, Customer27 es la clave primaria para un cliente específico. La salida desde esta segunda consulta será una instancia del objeto CustomerDetails. La salida de la primera consulta es menos clara: puede devolver una lista de objetos CustomerDetails, o puede devolver una lista de URI para objetos CustomerDetails (a la que el cliente puede ir para una recuperación individual). Ambas convenciones son comunes.

Para implementar una recopilación en CICS, cree un esquema JSON que describa una lista de instancias de datos o una lista de URI. Puede entonces crear el servicio e implementarlo como normal. En este ejemplo, puede elegir implementar únicamente el método GET. Puede tener en cuenta la implementación de un servicio de paginación para permitir a un cliente avanzar y retroceder página a través de conjuntos de datos grandes, por ejemplo:

```
GET
/Services/CustomerDetails?startRecord=200&endRecord=225
```

Es probable que necesite dos recursos URIMAP en CICS (y dos recursos WEBSERVICE). Uno que correlacione la estructura URI raíz para la recopilación y URIMAP de carácter comodín para las instancias. Por ejemplo:

```
URIMAP1: Path=/Services/CustomerDetails
WEBSERVICE=CollectionService
URIMAP2: Path=/Services/CustomerDetails/* WEBSERVICE=InstanceService
```

Gestión de memoria caché

La arquitectura RESTful anima a la integración con técnicas de gestión de memoria caché HTTP estándar. Este permite que los resultados de las solicitudes GET se guarden en la memoria caché de la red, reduciendo así la carga en el servidor. El mecanismo para hacer esto implica el establecimiento de una fecha y hora de caducidad para los datos devueltos para las solicitudes GET.

No hay un mecanismo de propósito general para soportar la aplicación que ha controlado la caducidad de la memoria caché en CICS, pero se puede escribir un programa de manejador de interconexión para añadir la cabecera HTTP adecuada utilizando la API EXEC CICS WEB WRITE HTTPHEADER. Los programas de aplicación pueden hacer algo parecido, pero sólo si están alojados en la misma región de CICS que recibe la solicitud HTTP.

Matrices de variables de elementos en DFHJS2LS

JSON puede incluir matrices de distintos números de elementos. En general, los esquemas JSON que contienen varios números de elementos no se correlacionan de forma eficiente en una única estructura de datos de lenguaje de alto nivel. CICS utiliza correlaciones basadas en contenedores o correlaciones en línea para gestionar varios elementos en datos JSON.

Una matriz con varios elementos se representa en el esquema JSON utilizando las palabras clave `minItems` y `maxItems` en el esquema con el valor `"type"` de `"array"` :

- La palabra clave `minItems` especifica el número de veces mínimo que se puede dar un elemento. Puede tener un valor de 0 o un entero positivo. El valor predeterminado es 0.
- La palabra clave `maxItems` especifica el número de veces máximo que se puede dar un elemento. Puede tener un valor de entero positivo mayor o igual al valor de la palabra clave `minItems`.
- Si falta la palabra clave `maxItems`, significa que la matriz no está limitada.

Se puede indicar un campo opcional por medio de una matriz de variables `"maxItems":1` . Por ejemplo, una serie de 8 bytes opcional denominada `"component"` :

```
"properties":{
  "component": {
    "type":"array",
    "maxItems":1,
    "items": {
      "type": "string",
      "maxLength": 8
    }
  },
  "required": ["component"]
}
```

Se puede obtener el mismo efecto no incluyendo el nombre de campo en el valor de palabra clave `"required"`:

```
"properties":{
  "component": {
    "type": "string",
    "maxLength": 8
  }
}
```

En general, los esquemas JSON que contienen varios números de elementos no se correlacionan de forma eficiente en una única estructura de datos de lenguaje de alto nivel. Para gestionar estos casos, CICS utiliza una serie de estructuras de datos conectadas que se pasan al programa de aplicación en una serie de contenedores. Estas estructuras se utilizan como entrada y salida de la aplicación:

- Cuando CICS transforma datos JSON en datos de aplicación, llena estas estructuras con los datos de aplicación y la aplicación los lee.
- Cuando CICS transforma los datos de aplicación en datos JSON, lee los datos de aplicación en las estructuras que ha estado llenando la aplicación.

Los ejemplos siguientes ilustran el formato de estas estructuras de datos. Estos ejemplos utilizan una matriz de campos simples de 8 bytes. Sin embargo, el modelo soporta matrices de tipos de datos complejos y matrices de tipos de datos que contienen otras matrices.

Ejemplo 1. Número fijo de elementos

Este ejemplo ilustra un elemento que se produce tres veces exactamente:

```
"properties":{
  "component": {
    "type": "array",
    "maxItems": 3,
    "minItems": 3,
    "items": {

```

```

"type": "string",
"maxLength": 8
}
},
"required": ["component"]

```

En este ejemplo, el número de veces que se da un elemento se conoce de antemano, por lo tanto, se puede representar como una matriz de longitud fija en una sencilla declaración COBOL o el equivalente en otros lenguajes.

```
05 component PIC X(8) OCCURS 3 TIMES
```

Ejemplo 2. Número de elementos variable en el nivel de correlación 2 e inferior

Este ejemplo ilustra un elemento obligatorio que puede encontrarse de una a cinco veces:

```

"properties":{
  "component": {
    "type": "array",
    "maxItems": 5,
    "minItems": 1,
    "items": {
      "type": "string",
      "maxLength": 8
    }
  }
},
"required": ["component"]

```

La estructura de datos principal contiene una declaración de dos campos. Cuando CICS transforma los datos JSON en datos binarios, el primer campo component-num contiene el número de veces que aparece el elemento en los datos JSON, y el segundo campo, component-cont , contiene el nombre de un contenedor:

```
05 component-num PIC S9(9) COMP-5
05 component-cont PIC X(16)
```

Una segunda estructura de datos contiene la declaración del elemento:

```
01 DFHJS-component
02 component PIC X(8)
```

Debe examinar el valor de component-num , que contendrá un valor en el rango de 1 a 5, para averiguar cuántas veces se da el elemento. El contenido del elemento está en el contenedor denominado component-cont; el contenedor tiene una matriz de elementos, donde cada elemento está correlacionado por la estructura de datos DFHJS-component.

Si minItems="0" o falta y maxItems="1" , el elemento es opcional. Para procesar la estructura de datos en el programa de aplicación, debe examinar el valor de component-num:

- Si es cero, el mensaje no tiene elemento de componente y el contenido de component-cont no está definido.
- Si es uno, el elemento de componente está en el contenedor denominado component-cont.

El contenido del contenedor está correlacionado por la estructura de datos DFHJS-component.

Nota: Si los datos JSON constan de un único elemento recurrente, DFHJS2LS genera dos estructuras de lenguaje. La estructura de lenguaje principal contiene el número de elementos en la matriz y el nombre del contenedor que tiene la matriz de elementos. La segunda estructura de lenguaje correlaciona una única instancia de elemento recurrente.

Ejemplo 3. Número variable de elementos a nivel de correlación 2.1 y superior

En el nivel de correlación 2.1 y superior, puede utilizar el parámetro **INLINE-MAXOCCURS-LIMIT** en los asistentes de CICS. El parámetro **INLINE-MAXOCCURS-LIMIT** especifica la forma en la que se manejan varios elementos. Las opciones de correlación para un número variado de elementos son correlaciones basadas en contenedores, descritas en “Ejemplo 2. Número de elementos variable en el nivel de correlación 2 e inferior” en la página 285, o correlaciones en línea. El valor de este parámetro puede ser un entero positivo en el rango de 0 a 32767:

- El valor predeterminado de **INLINE-MAXOCCURS-LIMIT** es 1, que garantiza que los elementos opcionales están correlacionados en línea.
- Un valor de 0 para el parámetro **INLINE-MAXOCCURS-LIMIT** impide la correlación en línea.
- Si `maxItems` es menor o igual al valor de **INLINE-MAXOCCURS-LIMIT**, se utiliza la correlación en línea.
- Si `maxItems` es mayor que el valor de **INLINE-MAXOCCURS-LIMIT**, se utiliza la correlación basada en contenedores.

La correlación de un número variado de elementos en línea da como resultado la generación de una matriz, como sucede con el ejemplo de aparición fija anterior, y un contador. El campo `component-num` indica cuántas instancias del elemento están presentes y si están señaladas por una matriz. Para el ejemplos mostrado en “Ejemplo 2. Número de elementos variable en el nivel de correlación 2 e inferior” en la página 285, cuando **INLINE-MAXOCCURS-LIMIT** es menor o igual a 5, la estructura de datos generada es como ésta:

```
05 component-num PIC S9(9) COMP-5 SYNC.  
05 component OCCURS 5 PIC X(8).
```

El primer campo, `component-num`, es idéntico a la salida para el ejemplo de correlación basada en contenedores de la sección anterior. El segundo campo contiene una matriz de longitud 5 que es lo suficientemente larga para contener el número máximo de elementos que puede generar.

La correlación en línea difiere de la correlación basada en contenedores, que almacena el número de apariciones del elemento y el nombre del contenedor donde están colocados los datos, porque almacena todos los datos en el contenedor actual. Almacenar los datos en el contenedor actual normalmente mejorará el rendimiento y hará que la correlación en línea sea preferible.

Ejemplo 4. Matrices de variables anidadas

Los esquemas JSON complejos pueden contener elementos recurrentes variables, que a su vez contienen elementos recurrentes variables. En este caso, la estructura descrita se amplía más allá de los dos niveles descritos en los ejemplos.

Este ejemplo ilustra un elemento opcional llamado "component2" que se anida en un elemento obligatorio llamado "component1" , donde el elemento obligatorio se puede dar de una a cinco veces:

```
"properties":{
  "component1": {
    "type": "array",
    "maxItems": 5,
    "minItems": 1,
    "items": {
      "type": "object",
      "properties":{
        "component2":{
          "type": "string",
          "maxLength": 8
        }
      },
      "required": ["component2"]
    }
  },
  "required": ["component1"]
}
```

La estructura de datos de nivel superior es exactamente la misma que en los ejemplos anteriores:

```
05 component1-num PIC S9(9) COMP-5
05 component1-cont PIC X(16)
```

Sin embargo, la segunda estructura de datos contiene estos elementos:

```
01 DFHJS-component1
02 component2-num PIC S9(9) COMP-5
02 component2-cont PIC X(16)
```

Una estructura de tercer nivel contiene estos elementos:

```
01 DFHJS-component2
02 component2 PIC X(8)
```

El número de apariciones del elemento más externo "component1" está en component1-num.

El contenedor nombrado en component1-cont contiene una matriz con varias instancias de la segunda estructura de datos DFHJS-component1.

Cada instancia de component2-cont nombra un contenedor diferente, cada uno de los cuales contiene la estructura de datos correlacionada por la estructura de tercer nivel DFHJS-component2.

Para ilustrar esta estructura, tenga en cuenta el fragmento de datos JSON que coincide con el ejemplo:

```
{ "component1":
[
{
  "component2": "string1"
},
{
  "component2": "string2"
},
]
}
```

"component1" se da tres veces. Las dos primeras contienen una instancia de "component2" ; la tercera instancia no.

En la estructura de datos de nivel superior, component1-num contiene un valor de 3. El contenedor nombrado en component1-cont tiene tres instancias de DFHJS-component1 :

1. En la primera, component2-num tiene un valor de 1, y el contenedor nombrado en component2-cont contiene *string1*.
2. En la segunda, component2-num tiene un valor de 1, y el contenedor nombrado en component2-cont contiene *string2*.
3. En la tercera, component2-num tiene un valor de 0, y el contenido de component2-cont no está definido.

En esta instancia, la estructura de datos completa se representa por medio de cuatro contenedores en total:

- La estructura de datos raíz en el contenedor DFHJS-DATA.
- El contenedor nombrado en component1-cont.
- Los dos contenedores nombrados en las dos primeras instancias de component2-cont.

Estructuras opcionales y la palabra clave **required**

Las estructuras de datos se definen por el esquema JSON "type" de "object". Los esquemas relacionan nombres de campos con tipos individuales utilizando el objeto proporcionado por la palabra clave "properties". El requisito para que estos campos sean parte de los datos JSON descritos por el esquema JSON se controla por medio de la matriz proporcionada por la palabra clave "required". Esta matriz lista todos los nombres de campo que deben estar presentes en los datos JSON. Por lo tanto, los campos opcionales están representados por su ausencia de la matriz, o cuando la matriz no puede estar vacía, la ausencia de la palabra clave "required". En este caso, todos los campos son opcionales.

Los campos opcionales se tratan como matriz variable de 0 o 1 elementos. Esto añade un campo adicional con el sufijo "-num" añadido al nombre de elemento. Si la longitud total es mayor de 28 caracteres, el nombre de elemento se trunca. En el tiempo de ejecución, no será cero para indicar que el valor estaba presente en los datos JSON y cero si el valor no lo está.

Este ejemplo muestra dos campos, uno necesario llamado "required-structure" y el otro opcional llamado "optional-structure" :

```
{
  "type": "object",
  "properties": {
    "required-structure": {
      "type": "string",
      "maxLength": 8
    },
    "optional-structure": {
      "type": "string",
      "maxLength": 8
    }
  },
  "required": [
    "required-structure"
  ]
}
```

La estructura COBOL generada muestra ambos campos, pero la segunda va precedida de "optional-structure-num" que es un recuento de enteros de los

elementos, con 0 representando a ninguno y 1 que está presente. El valor se establece para indicar si "optional-structure" contiene datos válidos o no.

03 OutputData.

06 required-structure PIC X(8).

06 optional-structure-num PIC S9(9) COMP-5 SYNC.

06 optional-structure PIC X(8).

Creación de informes de errores de aplicación

Lea este tema para comprender cómo las aplicaciones del proveedor de servicios web JSON RESTful pueden informar de errores a los clientes.

En escenarios descendentes, es probable que se necesite la aplicación para informar de condiciones de error. Por ejemplo, un error puede ser: " Número de cuenta no reconocido ". Este requisito es exclusivo para escenarios descendentes, en un desarrollo ascendente, la aplicación tiene un mecanismo de creación de informes de error que está codificado en campos de datos o termina anormalmente. En el escenario descendente, no es probable que el esquema JSON defina un campo en el que informar de errores, así que es necesaria una alternativa.

Para servicios web basados en SOAP, este problema se trata utilizando la API **EXEC CICS SOAPFAULT**. Los mensajes de error SOAP no existen en JSON. En lugar de eso, puede utilizar el contenedor DFHHTTPSTATUS para informar de los errores detectados por la aplicación para aplicaciones JSON. Para obtener más información, consulte el apartado DFHHTTPSTATUS.

Nota: Las aplicaciones también pueden utilizar el contenedor DFHRESPONSE y, otros contenedores de control, para proporcionar una respuesta de error detallada, si desean hacerlo.

Restricciones de servicio web JSON

Utilice este material de referencia para comprender las prestaciones no compatibles con los servicios web JSON.

Las siguientes posibilidades no están soportadas:

- No se admiten las interconexiones en modo de solicitante con JSON.
- No se admite la validación en tiempo de ejecución de los datos JSON con el esquema. Se omite el valor del atributo VALIDATION de un recurso WEBSERVICE que se utiliza con una carga útil JSON.
- No se admite el uso de espacios de nombres en datos JSON (convenciones Badgerfish o correlacionadas).
- Las cargas útiles de JSON enviadas a CICS deben estar codificadas en UTF-8. No se admite ninguna otra codificación. De forma similar, el JSON enviado a CICS está siempre codificado en UTF-8.
- No se admiten transportes WebSphere MQ con interconexiones JSON.
- No se admiten programas transformadores de proveedor para su uso con el transformador de JSON.
- No se admite la reutilización de archivos WSBind creados para aplicaciones de servicios web SOAP un una interconexión JSON. El asistente de JSON debe generar los archivos WSBind para su uso con las aplicaciones de proveedor de servicios JSON.
- Si a una carga útil de JSON le falta contenido obligatorio cuando CICS lo transforma, los campos equivalentes dentro de las estructuras de datos no se inicializan cuando pasan al programa de aplicación.

- CICS no puede transformar valores enteros mayores que el valor máximo para una serie larga firmada ($2^{63} - 1$) a menos que esté encerrada entre comillas.
- No se admite el uso de tipos de datos simples en la raíz de un esquema JSON. El esquema JSON describe un objeto JSON o una matriz JSON, aunque el objeto JSON puede, a su vez, contener tipos de datos simples, matrices y otros objetos.
- Si se declara una matriz en un esquema JSON con un valor **maxItems** de 1, CICS serializa la matriz como un entero o serie simple cuando genera JSON en tiempo de ejecución.

Importante: Los únicos caracteres soportados para los nombres de propiedad de JSON son: A-Z a-z _ : para el primer carácter y A-Z a-z 0-9 _ : . - para los siguientes caracteres.

El soporte de servicios web Axis2 tiene hoy día varias opciones para el desarrollo y despliegue de aplicaciones y personalizaciones. Los valores siguientes no están soportados:

- Manejadores de aplicación proporcionados por el usuario - debe utilizar la clase de manejador de aplicación proporcionada por CICS
`com.ibm.cicsts.axis2.CICSAxis2ApplicationHandler`.
- Aplicaciones Axis2 de Java escritas por el usuario.
- Las API de SOAPFAULT y WS-Addressing no se pueden utilizar con la interconexión JSON.

Restricciones de contenedor

Nota: Algunos contenedores de interconexión no se llenan cuando se procesa una solicitud JSON. Para obtener más información, consulte el apartado Containers used in the pipeline.

Diferencias en los servicios web RESTful

En INQUIRE PIPELINE:

- SOAPLEVEL devuelve NOTSOAP
- Los atributos MTOMNOXOPST, MTOMST, SENDMTOMST, SOAPRNUM, SOAPVNUM, XOPDIRECTST e XOPSUPPORTST no se utilizan.

En INQUIRE WEBSERVICE:

- Los atributos ARCHIVEFILE, BINDING, VALIDATIONST, XOPDIRECTST y XOPSUPPORTST no se utilizan.
- WSDLFILE devuelve el nombre del archivo de esquema JSON asociado a WEBSERVICE.

En el recurso WEBSERVICE:

- Los parámetros ARCHIVEFILE y VALIDATION no se utilizan y los valores se ignoran.
- WSDLFILE es el nombre del archivo de esquema JSON asociado a WEBSERVICE.

Correlación y transformación de datos de aplicación y XML

Puede escribir programas de aplicación para transformar datos binarios de la aplicación en XML y viceversa. CICS soporta varios lenguajes de alto nivel y proporciona un asistente XML para correlacionar cómo se transforman los datos durante el proceso de tiempo de ejecución. CICS utiliza la misma tecnología para la correlación de datos de aplicación a XML en mensajes SOAP, como parte del soporte de servicios web.

Antes de empezar

Debe tener instalado Java para ejecutar el asistente XML y para realizar la validación opcionalmente cuando CICS transforma los datos o XML.

Acerca de esta tarea

La ventaja de utilizar este enfoque para transformar datos de aplicación a y desde XML es que CICS va más allá de las prestaciones ofrecidas por un analizador XML. CICS puede interpretar el XML y realizar las conversiones basadas en registros de los datos de aplicación. Por lo tanto, es más fácil y más rápido crear aplicaciones que funcionen con XML utilizando este enfoque.

El asistente XML de CICS es un programa de utilidad proporcionado que le ayuda a crear los artefactos requeridos para transformar datos binarios de aplicación en XML o transformar XML en datos binarios de aplicación. El asistente XML puede crear los artefactos en un directorio de paquete y otra ubicación especificada en z/OS UNIX.

Procedimiento

1. Cree las correlaciones utilizando el asistente XML.
2. Cree los recursos XMLTRANSFORM en CICS para que las correlaciones estén disponibles.
3. Cree o actualice un programa de aplicación para utilizar el mandato de API **TRANSFORM**. La aplicación debe utilizar una interfaz basada en canal.
4. Ejecute la aplicación para probar que la transformación funciona como pretendía. Puede activar la validación para comprobar que CICS convierte los datos correctamente.

Qué hacer a continuación

Estos pasos se explican de forma más detallada en los temas siguientes.

El asistente XML de CICS

El asistente XML de CICS es un conjunto de programas de utilidad por lotes que pueden ayudarle a transformar XML en estructuras de lenguaje de alto nivel y viceversa. El asistente soporta un despliegue de aplicaciones rápido que lleva a cabo el procesamiento de XML con la mínima cantidad de esfuerzo de programación.

Utilizando el asistente XML para CICS se reduce la cantidad de código que debe escribir para analizar o construir XML; CICS transforma datos entre fragmentos de XML y la estructura de datos de un programa de aplicación.

El asistente XML puede crear un esquema XML desde una sencilla estructura de lenguaje, o una estructura de lenguaje desde un esquema XML existente y soporta COBOL, C/C++ y PL/I. También genera metadatos que utiliza CICS en el tiempo de ejecución para convertir automáticamente datos XML en datos de aplicación binarios o viceversa; los metadatos se definen en un enlace XML y se almacenan en z/OS UNIX. El esquema para el enlace XML está en el directorio /usr/lpp/cicsts/cicsts52/schemas/xmltransform/ en z/OS UNIX.

El asistente de XML de CICS consta de dos programas de utilidad:

DFHLS2SC

Este programa de utilidad genera un enlace y esquema XML desde una estructura de lenguaje.

DFHSC2LS

Este programa de utilidad genera una estructura de lenguaje y un enlace XML que puede utilizar en los programas de aplicación. Puede utilizar un documento WSDL o un esquema XML como entrada.

Los procedimientos JCL para ejecutar ambos programas están en la biblioteca *hlq*.XDFHINST, donde *hlq* es el calificador de alto nivel de la instalación CICS.

DFHLS2SC: Conversión de lenguaje de alto nivel a esquema XML

El procedimiento catalogado DFHLS2SC genera un esquema XML y un archivo de enlace XML desde una estructura de lenguaje de alto nivel. Utilice DFHLS2SC cuando desee crear un programa CICS que pueda analizar o crear XML. Se enumeran las sentencias de control de trabajos de DFHLS2SC, sus parámetros simbólicos, sus parámetros de entrada y sus descripciones.

Sentencias de control de trabajos para DFHLS2SC

JOB Inicia el trabajo.

EXEC Especifica el nombre de procedimiento (DFHLS2SC).

INPUT.SYSUT1 DD

Especifica la entrada. Los parámetros de entrada se especifican normalmente en el flujo de entrada. Sin embargo, se pueden definir en un conjunto de datos en miembro de un conjunto de datos particionados.

Parámetros simbólicos

Los siguientes parámetros simbólicos se definen en DFHLS2SC:

JAVADIR = *vía de acceso*

Especifica el nombre del directorio Java que utiliza DFHLS2SC. El valor de este parámetro se añade a /usr/lpp/ para crear un nombre de vía de acceso completo de /usr/lpp/ *vía de acceso*.

Generalmente, no es necesario especificar este parámetro. El valor predeterminado es el valor proporcionado para el trabajo de instalación (DFHISTAR) de CICS en el parámetro **JAVADIR**.

PATHPREF = *prefijo*

Especifica un prefijo opcional que amplía la vía de acceso del directorio z/OS UNIX utilizado en otros parámetros. El valor predeterminado es la serie vacía.

Generalmente, no es necesario especificar este parámetro. El valor predeterminado es el valor proporcionado para el trabajo de instalación (DFHISTAR) de CICS en el parámetro **PATHPREF**.

SERVICE = *valor*

Utilice este parámetro sólo cuando el soporte de IBM le indique que lo haga.

TMPDIR = *tmpdir*

Especifica la ubicación de un directorio en z/OS UNIX que DFHLS2SC utiliza como espacio de trabajo temporal. El ID de usuario bajo el que se ejecuta el trabajo debe tener permiso de lectura y escritura para este directorio.

El valor predeterminado es /tmp.

TMPFILE = *tmpprefix*

Especifica un prefijo que DFHLS2SC utiliza para construir los nombres de los archivos de espacio de trabajo temporal.

El valor predeterminado es SC2WS.

USSDIR = *vía de acceso*

Especifica el nombre del directorio CICS TS en el sistema de archivos z/OS UNIX. El valor de este parámetro se añade a /usr/lpp/cicsts/ para crear un nombre de vía de acceso completo de /usr/lpp/cicsts/ *vía de acceso* .

Generalmente, no es necesario especificar este parámetro. El valor predeterminado es el valor proporcionado para el trabajo de instalación (DFHISTAR) de CICS en el parámetro **USSDIR**.

Espacio de trabajo temporal

DFHLS2SC crea estos tres archivos temporales durante la ejecución:

```
tmpdir / tmpprefix .in  
tmpdir / tmpprefix .out  
tmpdir / tmpprefix .err
```

donde:

tmpdir es el valor que se especifica en el parámetro **TMPDIR**.

tmpprefix es el valor que se especifica en el parámetro **TMPFILE**.

Los nombres predeterminados para los archivos (cuando no se especifica **TMPDIR** y **TMPFILE** son los siguientes:

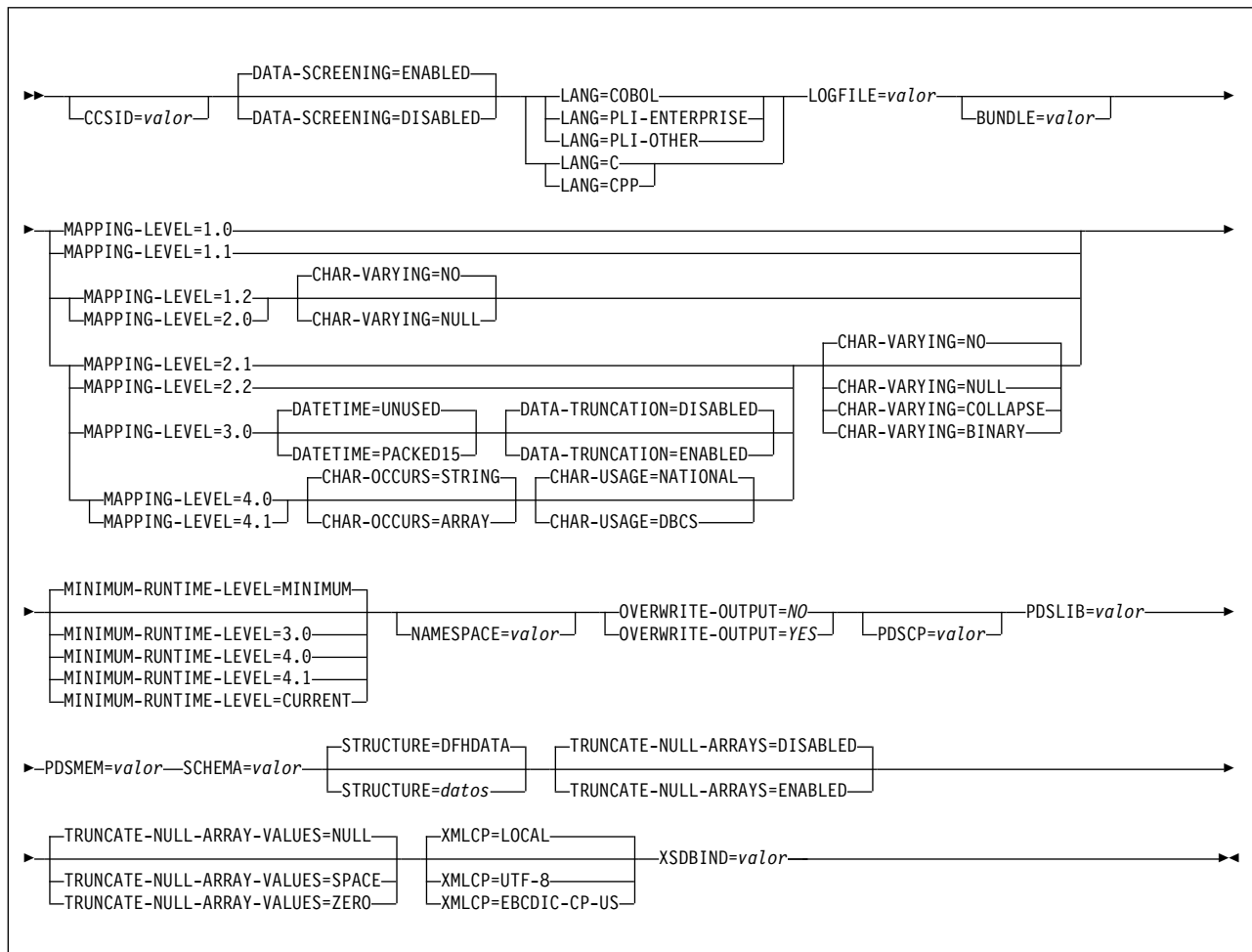
```
/tmp/LS2SC.in  
/tmp/LS2SC.out  
/tmp/LS2SC.err
```

Importante: DFHLS2SC no bloquea el acceso a los archivos z/OS UNIX o los miembros del conjunto de datos. Por lo tanto, si dos o más instancias de DFHLS2SC se ejecutan de forma simultánea y utilizan los mismos archivos de espacio de trabajo temporal, nada impide a un trabajo sobrescribir los archivos de espacio de trabajo mientras otro trabajo los está utilizando, llevando a errores imprevisibles.

Por lo tanto, se aconseja crear un convenio de denominación y procedimientos operativos que eviten esta situación; por ejemplo, puede utilizar el parámetro simbólico de sistema **SYSUID** para generar nombres de archivo de espacios de trabajo que son exclusivos para un usuario individual.

Estos archivos temporales se suprimen antes de la finalización del trabajo.

Parámetros de entrada de DFHLS2SC



Utilización de parámetros

- Puede especificar los parámetros de entrada en cualquier orden.
- Cada parámetro se debe iniciar en una nueva línea.
- Un parámetro (y su carácter de continuación, si utiliza uno) no deben sobrepasar la columna 72; las columnas 73 a 80 deben contener espacios en blanco.
- Si un parámetro es demasiado largo para ajustarse en una única línea, utilice el carácter de asterisco (*) al final de la línea para indicar que el parámetro continúa en la línea siguiente. Todo (incluidos los espacios) antes del asterisco se considera parte del parámetro. Por ejemplo:

```
XSDBIND=/path/xsdbindir*
      /app1
```

es equivalente a

```
XSDBIND=/path/xsdbindir/app1
```

- Un carácter # en la primera posición de carácter de la línea es un carácter de comentario. La línea se ignora.
- Una coma en la última posición de carácter de la línea es un separador de línea opcional y se ignora.

Descripciones de parámetros

BUNDLE = *valor*

Especifica la vía de acceso y el nombre del directorio del paquete en z/OS UNIX. Si especifica este valor, el asistente XML genera un paquete que contiene un enlace XSD. La información de vía de acceso en este parámetro sustituye cualquier información de vía de acceso en el parámetro **XSDBIND**.

Opcionalmente, puede especificar un archivo de archivado en lugar de un nombre de directorio. El asistente XML soporta archivos .zip y .jar. Sin embargo, debe descomprimir los archivos antes de intentar instalar el recurso BUNDLE.

Si no especifica este parámetro, CICS coloca el enlace y esquema XML en la ubicación especificada en el parámetro **XSDBIND**.

CCSID = *valor*

Especifica el CCSID que se utiliza en tiempo de ejecución para codificar datos de caracteres en la estructura de datos de la aplicación. El valor de este parámetro sustituye el valor del parámetro de inicialización del sistema **LOCALCCSID**. El *valor* debe ser un EBCDIC CCSID soportado por los servicios de conversión de Java y z/OS. Si no especifica este parámetro, la estructura de datos de aplicación se codifica utilizando el CCSID especificado en el parámetro de inicialización del sistema.

Puede utilizar este parámetro con cualquier nivel de correlación.

CHAR-VARYING = { **NO** | **NULL** | **COLLAPSE** | **BINARY** }

Especifica cómo los campos de carácter de la estructura de lenguaje se correlacionan cuando el nivel de correlación es 1.2 o superior. Un campo de carácter en COBOL es una cláusula Picture de tipo X; por ejemplo, PIC(X) 10 . Un campo de caracteres en C/C++ es una matriz de caracteres. Este parámetro no se aplica a estructuras de lenguaje Enterprise y Otros PL/I. Puede seleccionar estas opciones:

NO Los campos de carácter se correlacionan con un `xsd:string` y se procesan como campos de longitud fija. La longitud máxima de los datos es igual a la longitud del campo. NO es el valor predeterminado para el parámetro **CHAR-VARYING** para COBOL y PL/I a niveles de correlación 2.0 y anteriores.

NULL Los campos de carácter se correlacionan con un `xsd:string` y se procesan como series terminadas en nulo. CICS añade un carácter terminado en nulo cuando se está transformando desde un mensaje XML. La longitud máxima de la serie de caracteres se calcula como un carácter menos que la longitud indicada en la estructura de lenguaje. NULL es el valor predeterminado para el parámetro **CHAR-VARYING** para C/C++.

COLLAPSE

Los campos de carácter se correlacionan con una `xsd:string` . Los espacios en blanco finales del campo no se incluyen en el esquema XML. COLLAPSE es el *valor* predeterminado para el parámetro **CHAR-VARYING** para COBOL y PL/I en niveles de correlación 2.1 y en adelante.

BINARY

Los campos de carácter se correlacionan como un tipo de datos `xsd:base64binary` y se procesan como campos de longitud fija. El *valor* BINARY en el parámetro **CHAR-VARYING** está disponible únicamente en niveles de correlación 2.1 y en adelante.

CHAR-OCCURS = { STRING | **ARRAY** }

Especifica cómo las matrices de caracteres en la estructura de lenguaje se correlacionan cuando el nivel de correlación es 4.0 o superior. Por ejemplo, PIC X OCCURS 20 . Este parámetro sólo lo utiliza el lenguaje COBOL.

ARRAY

Las matrices de caracteres se correlacionan con una matriz de XML. Esto significa que cada carácter se correlaciona como un elemento XML individual. Este también es el comportamiento en el nivel de correlación 3.0 y anterior.

STRING

Las matrices de caracteres se correlacionan con una serie XML. Esto significa que toda la matriz COBOL se correlaciona como un único elemento XML.

CHAR-USAGE = { NATIONAL | **DBCS** }

En COBOL, el tipo de datos nacional, PIC N , se puede utilizar para datos UTF-16 o DBCS. Este valor lo controla la opción del compilador NSYMBOL. Debe establecer el parámetro **CHAR-USAGE** en el asistente en el mismo valor que la opción de compilador NSYMBOL para asegurarse de que los datos se manejan adecuadamente. Se establece normalmente en CHAR-USAGE=NATIONAL cuando utiliza UTF-16.

DBCS Los datos de los campos PIC (*n*) se tratan como datos cifrados en DBCS.

NATIONAL

Los datos de los campos PIC (*n*) se tratan como datos cifrados en UTF-16.

DATA-SCREENING = { ENABLED | **DISABLED** }

Especifica si los datos proporcionados por la aplicación se analizan para detectar errores.

ENABLED

Los datos de tiempo de ejecución proporcionados por la aplicación que son incoherentes con la estructura de lenguaje, se tratan como un error y se emite un mensaje DFHPI1010. Se devuelve una respuesta de error a la aplicación.

DISABLED

Los valores de datos de los datos de tiempo de ejecución proporcionados por la aplicación que son incoherentes con la estructura de lenguaje se sustituyen por valores predeterminados. Por ejemplo, un cero sustituye un valor erróneo en un campo numérico. No se emite mensaje DFHPI1010 y se devuelve una respuesta normal a la aplicación. Esta característica se puede utilizar para evitar las respuestas de error INVALID_PACKED_DEC y INVALID_ZONED_DEC generadas desde los campos de salida no inicializados.

DATA-TRUNCATION = { DISABLED | **ENABLED** }

Especifica si los datos de longitud variable se soportan en una estructura de campo de longitud fija:

DISABLED

Si los datos son menores que la longitud fija que está esperando CICS, CICS rechaza los datos truncados y emite un mensaje de error.

ENABLED

Si los datos son menores que la longitud fija que está esperando, CICS soporta los datos truncados y procesa los datos que faltan como valores nulos.

DATETIME = { **UNUSED** | **PACKED15** }

Especifica si los campos ABSTIME potenciales en la estructura de lenguaje de alto nivel se correlacionan como indicaciones de fecha y hora:

PACKED15

Los campos decimales empaquetados de longitud 15 (8 bytes) se tratan como campos CICS ABSTIME y se correlacionan como indicaciones de fecha y hora.

UNUSED

Los campos decimales empaquetados de longitud 15 (8 bytes) no se tratan como indicaciones de fecha y hora.

Este parámetro se puede definir en un nivel de correlación de 3.0.

LANG = **COBOL**

Especifica que el lenguaje de programación de la estructura de lenguaje de alto nivel es COBOL.

LANG = **PLI-ENTERPRISE**

Especifica que el lenguaje de programación de la estructura de lenguaje de alto nivel es Enterprise PL/I.

LANG = **PLI-OTHER**

Especifica que el lenguaje de programación de la estructura de lenguaje de alto nivel es un nivel de PL/I distinto a Enterprise PL/I.

LANG = **C**

Especifica que el lenguaje de programación de la estructura de lenguaje de alto nivel es C.

LANG = **CPP**

Especifica que el lenguaje de programación de la estructura de lenguaje de alto nivel es C++.

LOGFILE = *valor*

El nombre completo del archivo de z/OS UNIX en el que DFHLS2SC escribe la información de rastreo y registro de actividad. DFHLS2SC crea el archivo pero no la estructura de directorio, si no existe aún.

Normalmente, no utiliza este archivo, pero puede solicitarlo la organización de servicio de IBM si encuentra problemas con DFHLS2SC.

MAPPING-LEVEL = { **1.0** | **1.1** | **1.2** | **2.0** | **2.1** | **2.2** | **3.0** | **4.0** | **4.1** }

Especifica el nivel de correlación para que el asistente lo utilice al generar las estructuras de lenguaje y enlace XML. Se recomienda utilizar el nivel de correlación más reciente que esté disponible. Si está creando un enlace XML para un feed Atom, debe utilizar un nivel de correlación 3.0.

MINIMUM-RUNTIME-LEVEL = { **MINIMUM** | **3.0** | **4.0** | | **4.1 CURRENT** }

Especifica el entorno de tiempo de ejecución de CICS mínimo en el que se puede desplegar el enlace XML. Si selecciona un nivel que no coincide con los demás parámetros que ha especificado, recibe un mensaje de error. Las opciones que puede seleccionar son las siguientes:

MINIMUM

El nivel de tiempo de ejecución más bajo posible de CICS se asigna automáticamente basándose en los parámetros que ha especificado.

- 3.0** Especifique el nivel de tiempo de ejecución 3.0 o superior si desea utilizar el asistente XML de CICS y aprovechar las correlaciones de datos avanzadas.
- 4.0** El archivo de enlace de servicios web generado se despliega correctamente en una región de CICS TS 5.2 o posterior. Con este nivel de ejecución, puede utilizar un nivel de correlación de 4.0 o anterior para el parámetro **MAPPING-LEVEL**. Puede utilizar cualquier parámetro opcional en este nivel.
- 4.1** El archivo de enlace de servicios web generado se despliega correctamente en una región de CICS V5.2 que tiene aplicado APAR PI67641, o en cualquier región de CICS V5.3 o posterior. Con este nivel de ejecución, puede utilizar un nivel de correlación de 4.1 o anterior para el parámetro MAPPING-LEVEL.

CURRENT

Utilice este nivel de tiempo de ejecución para desplegar el archivo de enlace generado en una región de CICS que tiene el conjunto de entorno de ejecución establecido en uno utilizado para generar el archivo de enlace.

NAMESPACE = *valor*

Especifica el espacio de nombres para que CICS lo utilice en el esquema XML generado. Para feed Atom, CICS proporciona este espacio de nombres en el feed Atom con el espacio de nombres Atom.

Si no especifica este parámetro, CICS genera un espacio de nombres automáticamente.

OVERWRITE-OUTPUT = **NO** | **YES**

Controla si los CICS BUNDLES existentes en el sistema de archivos se pueden sobrescribir.

NO No se sustituye ningún BUNDLE existente. Si un BUNDLE existente se encuentra DFHLS2SC emite un mensaje de error DFHPI9689E y finaliza.

YES Se sustituye cualquier BUNDLE existente. Si se encuentra un BUNDLE se emite el mensaje DFHPI9683W para informarle de que el archivo se ha sustituido.

PDSCP = *valor*

Especifica la página de códigos que se utiliza en los miembros de un conjunto de datos particionados, donde *valor* es un número CCSID o un número de página de códigos Java. Si no especifica este parámetro, se utiliza la página de códigos de los servicios del sistema z/OS UNIX. Por ejemplo, puede especificar PDSCP=037.

PDSLIB = *valor*

Especifica el nombre del conjunto de datos particionados que contiene las estructuras de datos de lenguaje de alto nivel que se van a procesar.

Restricción: Los registros del conjunto de datos particionados debe tener una longitud fija de 80 bytes.

PDSMEM = *valor*

Especifica el nombre del miembro de conjunto de datos que contiene las estructuras de lenguaje de alto nivel que se van a procesar.

SCHEMA = *valor*

El nombre completo del archivo de z/OS UNIX en el que se escribe el esquema XML. El esquema XML se ajusta a la especificación WSDL 2.0. DFHLS2SC crea el archivo pero no la estructura de directorio, si no existe aún.

STRUCTURE = { **DFHDATA** | *datos* }

Nombre de la estructura de datos de alto nivel en C y C++. El valor predeterminado es DFHDATA.

TRUNCATE-NULL-ARRAYS = { **DISABLED** | **ENABLED** }

Especifica cómo se procesan las matrices estructuradas en el nivel de correlación 4.1 o superior. Si está habilitado, CICS intentará reconocer los registros vacíos dentro de una matriz (consulte TRUNCATE-NULL-ARRAY-VALUES para obtener más información sobre la identificación de registros vacíos). Si se detectan cinco registros de matriz vacíos consecutivos, la matriz se trunca en el primero de esos registros cuando se genera XML/JSON. Esta prestación de recorte sólo se habilita para matrices con contenido estructurado, las matrices de campos primitivos simples no están sujetos al recorte. El recorte de matrices puede dar como resultado una representación más precisa de los datos en JSON/XML, pero no sin riesgo. Si cinco registros de datos consecutivos se identifican de forma incorrecta como almacenamiento no inicializado (quizás porque contienen legítimamente valores bajos), se puede experimentar una pérdida de datos. Si TRUNCATE-NULL-ARRAYS está habilitado y TRUNCATE-NULL-ARRAY-VALUES no se ha establecido, se utiliza el valor predeterminado para TRUNCATE-NULL-ARRAY-VALUES.

TRUNCATE-NULL-ARRAY-VALUES = { **NULL** | **SPACE** | **ZERO** }

Especifica qué valores se tratan como vacíos para el proceso de TRUNCATE-NULL-ARRAYS en el nivel de correlación 4.1 o superior. De forma predeterminada, el valor nulo (0x00, o los valores bajos) se tratan como vacíos. Si todos los bytes de almacenamiento dentro de un registro de una matriz estructurada contienen nulos, entonces todo el registro se considera vacío. Se pueden especificar uno o más de los valores NULL, SPACE y ZERO en una lista separada por comas, donde NULL implica un carácter nulo (0x00), SPACE implica un SBCS EBCDIC Space (0x40) y ZERO implica un cero decimal con zona (0xF0). Cualquier combinación coincidente de los bytes seleccionados dentro un registro de matriz estructurado provocará que todo el registro se identifique como vacío. Si TRUNCATE-NULL-ARRAY-VALUES tiene un valor definido, entonces TRUNCATE-NULL-ARRAYS debe estar habilitado.

XSDBIND = *valor*

El nombre completo del enlace XSD de z/OS UNIX. DFHLS2SC crea el archivo pero no la estructura de directorio, si no existe aún. Si se especifica el parámetro BUNDLE, excluya la vía de acceso. La extensión de archivo es .xsdbind.

DFHSC2LS: Conversión de un esquema XML a lenguaje de alto nivel

El procedimiento catalogado DFHSC2LS genera una estructura de datos de lenguaje de alto nivel y un enlace XML desde un esquema XML o documento WSDL. Utilice DFHSC2LS cuando desee crear un programa CICS que pueda analizar o crear XML. Este tema lista las sentencias de control de trabajo, parámetros simbólicos, parámetros de entrada y sus descripciones DFHSC2LS.

Sentencias de control de trabajos para DFHSC2LS

JOB Inicia el trabajo.

EXEC Especifica el nombre de procedimiento (DFHSC2LS).

INPUT.SYSUT1 DD

Especifica la entrada. Los parámetros de entrada se especifican normalmente en el flujo de entrada. Sin embargo, se pueden definir en un conjunto de datos en miembro de un conjunto de datos particionados.

Parámetros simbólicos

Los siguientes parámetros simbólicos se definen en DFHSC2LS:

JAVADIR = *vía de acceso*

Especifica el nombre del directorio Java que utiliza DFHSC2LS. El valor de este parámetro se añade a /usr/lpp/ para crear un nombre de vía de acceso completo de /usr/lpp/ *vía de acceso* .

Generalmente, no es necesario especificar este parámetro. El valor predeterminado es el valor proporcionado para el trabajo de instalación (DFHISTAR) de CICS en el parámetro **JAVADIR**.

PATHPREF = *prefijo*

Especifica un prefijo opcional que amplía la vía de acceso del directorio z/OS UNIX utilizado en otros parámetros. El valor predeterminado es la serie vacía.

Generalmente, no es necesario especificar este parámetro. El valor predeterminado es el valor proporcionado para el trabajo de instalación (DFHISTAR) de CICS en el parámetro **PATHPREF**.

TMPDIR = *tmpdir*

Especifica la ubicación de un directorio en z/OS UNIX que DFHSC2LS utiliza como estación de trabajo temporal. El ID de usuario bajo el que se ejecuta el trabajo debe tener permiso de lectura y escritura para este directorio.

El valor predeterminado es /tmp.

TMPFILE = *tmpprefix*

Especifica un prefijo que DFHSC2LS utiliza para construir los nombres de los archivos de espacio de trabajo temporal.

El valor predeterminado es SC2LS.

USSDIR = *vía de acceso*

Especifica el nombre del directorio CICS TS en el sistema de archivos z/OS UNIX. El valor de este parámetro se añade a /usr/lpp/cicsts/ para crear un nombre de vía de acceso completo de /usr/lpp/cicsts/ *vía de acceso* .

Generalmente, no es necesario especificar este parámetro. El valor predeterminado es el valor proporcionado para el trabajo de instalación (DFHISTAR) de CICS en el parámetro **USSDIR**.

SERVICE = *valor*

Utilice este parámetro sólo cuando el soporte de IBM le indique que lo haga.

Espacio de trabajo temporal

DFHSC2LS crea estos tres archivos temporales durante la ejecución:

```
tmpdir / tmpprefix .in  
tmpdir / tmpprefix .out
```

tmpdir / tmpprefix .err

donde:

tmpdir es el valor que se especifica en el parámetro **TMPDIR**.

tmpprefix es el valor que se especifica en el parámetro **TMPFILE**.

Los nombres predeterminados para los archivos (cuando no se especifica **TMPDIR** y **TMPFILE** son los siguientes:

/tmp/SC2LS.in

/tmp/SC2LS.out

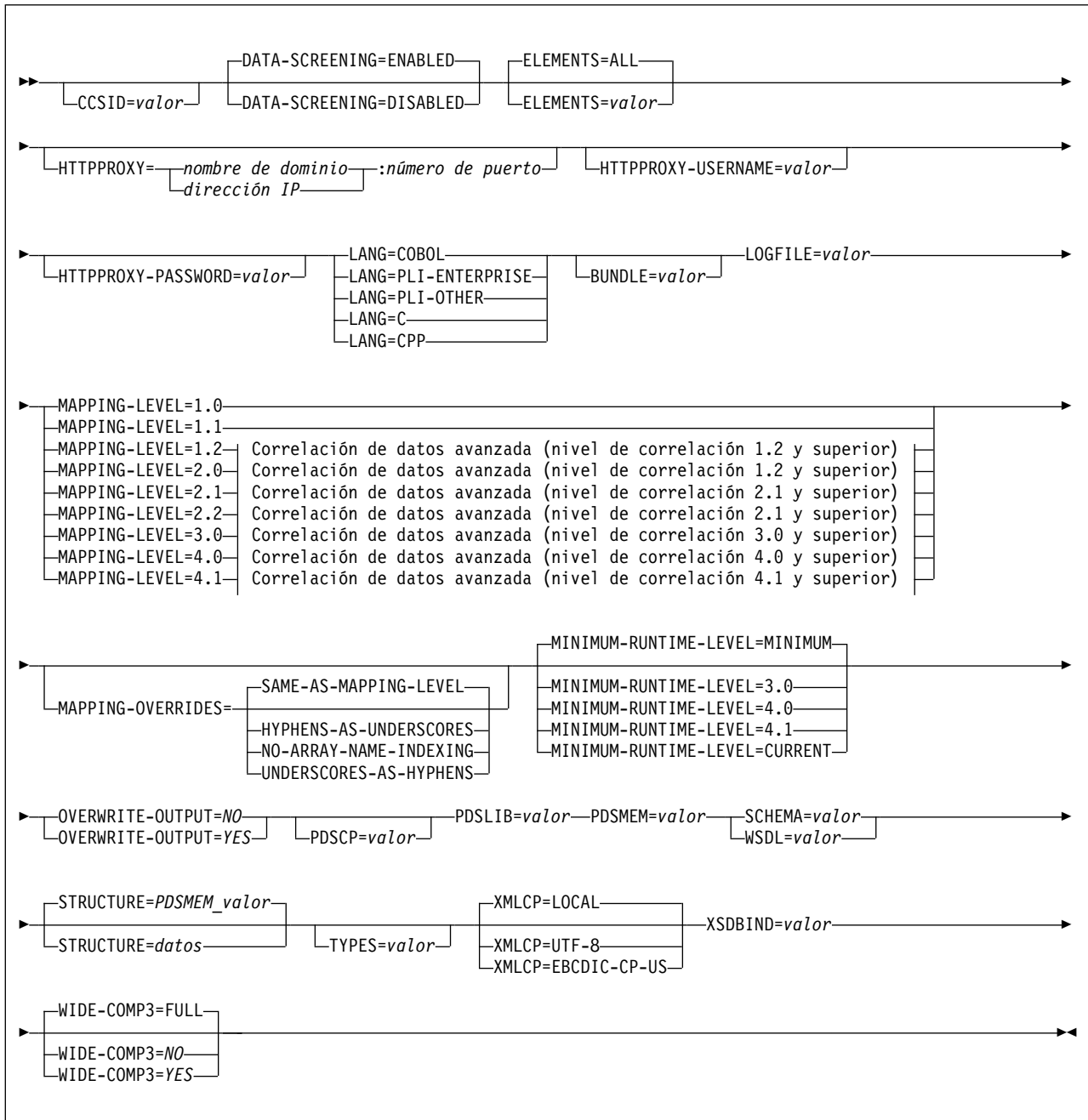
/tmp/SC2LS.err

Importante: DFHSC2LS no bloquea el acceso a los archivos z/OS UNIX o los miembros del conjunto de datos. Por lo tanto, si dos o más instancias de DFHSC2LS se ejecutan de forma simultánea y utilizan los mismos archivos de espacio de trabajo temporal, nada impide a un trabajo sobrescribir los archivos de espacio de trabajo mientras otro trabajo los está utilizando, llevando a errores imprevisibles.

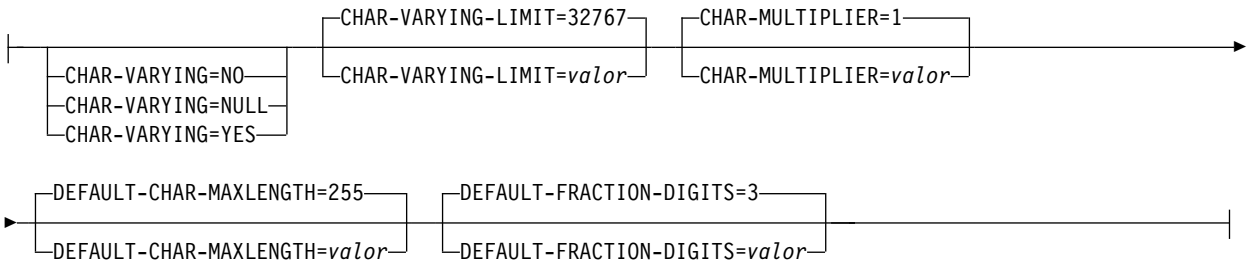
Por lo tanto, se aconseja crear un convenio de denominación y procedimientos operativos que eviten esta situación; por ejemplo, puede utilizar el parámetro simbólico de sistema **SYSUID** para generar nombres de archivo de espacios de trabajo que son exclusivos para un usuario individual.

Estos archivos temporales se suprimen antes de la finalización del trabajo.

Parámetros de entrada de DFHSC2LS



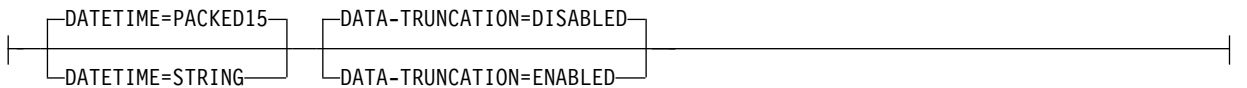
Correlación de datos avanzada (nivel de correlación 1.2 y superior):



Correlación de datos avanzada (nivel de correlación 2.1 y superior):



Correlación de datos avanzada (nivel de correlación 3.0):



Utilización de parámetros

- Puede especificar los parámetros de entrada en cualquier orden.
- Cada parámetro se debe iniciar en una nueva línea.
- Un parámetro (y su carácter de continuación, si utiliza uno) no deben sobrepasar la columna 72; las columnas 73 a 80 deben contener espacios en blanco.
- Si un parámetro es demasiado largo para ajustarse en una única línea, utilice el carácter de asterisco (*) al final de la línea para indicar que el parámetro continúa en la línea siguiente. Todo (incluidos los espacios) antes del asterisco se considera parte del parámetro. Por ejemplo:

```
XSDBIND=xsbinddir*  
/app1
```

es equivalente a

```
XSDBIND=xsbinddir/app1
```

- Un carácter # en la primera posición de carácter de la línea es un carácter de comentario. La línea se ignora.

Descripciones de parámetros

CCSID = *valor*

Especifica el CCSID que se utiliza en tiempo de ejecución para codificar datos de caracteres en la estructura de datos de la aplicación. El valor de este parámetro sustituye el valor del parámetro de inicialización del sistema **LOCALCCSID**. El *valor* debe ser un EBCDIC CCSID soportado por los servicios de conversión Java y z/OS (consulte z/OS Unicode Services User's Guide and

Reference). Si no especifica este parámetro, la estructura de datos de aplicación se codifica utilizando el CCSID especificado en el parámetro de inicialización del sistema.

Puede utilizar este parámetro con cualquier nivel de correlación.

CHAR-MULTIPLIER = { 1 | *valor* }

Especifica el número de bytes que se permiten para cada carácter cuando el nivel de correlación es 1.2 o posterior. El *valor* de este parámetro puede ser un entero positivo en el rango de 1 a 2.147.483.647. Todas las correlaciones basadas en caracteres no numéricos están sujetas a este multiplicador. Los campos binario, numérico, con zona y decimal empaquetado no están sujetos a este multiplicador.

Este parámetro puede ser útil si, por ejemplo, está planificando utilizar caracteres DBCS donde puede optar por un multiplicador de 3 para permitir espacio para caracteres de cambio y desplazamiento potenciales alrededor de todos los caracteres de doble byte en tiempo de ejecución.

Cuando establece **CCSID=1200** (indicando UTF-16), los únicos valores válidos para **CHAR-MULTIPLIER** son 2 o 4. Cuando utiliza UTF-16, el valor predeterminado es 2. Utilice **CHAR-MULTIPLIER=2** cuando espera que los datos de aplicación contengan caracteres que requieren 1 unidad de codificación UTF-16. Utilice **CHAR-MULTIPLIER=4** cuando espera que los datos de aplicación contengan caracteres que requieren 2 unidades de codificación UTF-16.

Nota: Establecer **CHAR-MULTIPLIER** en 1 no excluye el uso de caracteres DBCS, y establecerlo en 2 no excluye el uso de pares de sustitución UTF-16. Sin embargo, si los caracteres amplios utilizan de forma rutinaria algunos valores válidos no se ajustarán al campo asignado. Si se utiliza un valor **CHAR-MULTIPLIER** mayor, es posible almacenar más caracteres en el campo asignado que los que son válidos en el XML. Se debe tener cuidado para ajustarse a las restricciones de rango adecuadas.

CHAR-VARYING = { **NO** | **NULL** | **YES** }

Especifica cómo se correlacionan datos de caracteres de longitud variable cuando el nivel de correlación es 1.2 o superior. Los tipos de datos binarios de longitud variable se correlacionan siempre con un contenedor o una estructura variable. Si no especifica este parámetro, la correlación predeterminada depende del lenguaje especificado. Puede seleccionar estas opciones:

- NO** Los datos de caracteres de longitud variable se correlacionan como series de longitud fija.
- NULL** Los datos de caracteres de longitud variable se correlacionan con series terminadas en nulo.
- YES** Los datos de caracteres de longitud variable se correlacionan con un tipo de datos CHAR VARYING en PL/I. En los lenguajes COBOL, C y C++, los datos de caracteres de longitud variable se correlacionan con una representación equivalente que consta de dos elementos relacionados: la longitud de datos y los datos.

CHAR-VARYING-LIMIT = { 32767 | *valor* }

Especifica el tamaño máximo de datos binarios y de datos de caracteres de longitud variable que se correlacionan con la estructura de lenguaje cuando el nivel de correlación es 1.2 o superior. Si los datos binarios o de caracteres son mayores que el valor especificado en este parámetro, se correlacionan con un contenedor y el nombre de contenedor se utiliza en la estructura de lenguaje generada. El *valor* puede ir de 0 al valor predeterminado de 32.767 bytes.

DATA-SCREENING = { **ENABLED** | **DISABLED** }

Especifica si los datos proporcionados por la aplicación se analizan para detectar errores.

ENABLED

Los datos de tiempo de ejecución proporcionados por la aplicación que son incoherentes con la estructura de lenguaje, se tratan como un error y se emite un mensaje DFHPI1010. Se devuelve una respuesta de error a la aplicación.

DISABLED

Los valores de datos de los datos de tiempo de ejecución proporcionados por la aplicación que son incoherentes con la estructura de lenguaje se sustituyen por valores predeterminados. Por ejemplo, un cero sustituye un valor erróneo en un campo numérico. No se emite mensaje DFHPI1010 y se devuelve una respuesta normal a la aplicación. Esta característica se puede utilizar para evitar las respuestas de error INVALID_PACKED_DEC y INVALID_ZONED_DEC generadas desde los campos de salida no inicializados.

DATA-TRUNCATION = { **DISABLED** | **ENABLED** }

Especifica si los datos de longitud variable se soportan en una estructura de campo de longitud fija:

DISABLED

Si los datos son menores que la longitud fija que está esperando CICS, CICS rechaza los datos truncados y emite un mensaje de error.

ENABLED

Si los datos son menores que la longitud fija que está esperando, CICS soporta los datos truncados y procesa los datos que faltan como valores nulos.

DATETIME = { **PACKED15** | **STRING** }

Especifica que los campos xsd:dateTime están correlacionados con el formato de datos ABSTIME de CICS o con el texto:

PACKED15

Los campos xsd:dateTime están correlacionados con el formato ABSTIME de CICS.

STRING

Los campos xsd:dateTime se correlacionan con texto. Esta correlación es la misma que todos los niveles de correlación anteriores.

Este parámetro se puede utilizar en un nivel de correlación de 3.0.

DEFAULT-CHAR-MAXLENGTH = { **255** | *valor* }

Especifica la longitud de matriz predeterminada de los datos de caracteres en caracteres para correlaciones donde no está implicada la longitud en el documento de esquema XML o documento WSDL, cuando el nivel de correlación es 1.2 o superior. El *valor* de este parámetro puede ser un entero positivo en el rango de 1 a 2 147 483 647.

DEFAULT-FRACTION-DIGITS = **3** | *valor*

Especifica el número de dígitos de fracción predeterminado a utilizar en un tipo de esquema decimal XML. El valor predeterminado es 3. Para COBOL, el rango válido es 0-17, o 0-30 si se está utilizando **WIDE-COMP3**. Para C o PL/I, el rango válido es 0-30.

ELEMENTS = { ALL | *valor* }

Define una lista de nombres locales de elemento global para habilitar. El valor predeterminado de ALL indica que están habilitados todos los elementos globales.

HTTPPROXY = { *nombre de dominio* | *dirección IP* } : *número de puerto*

Si el esquema XML o documento WSDL contiene referencias a otro esquema XML o archivos WSDL ubicados en Internet, y el sistema en el que está ejecutando DFHSC2LS utiliza un servidor proxy para acceder a Internet, especifique el nombre de dominio o dirección IP y el número de puerto del servidor proxy. Por ejemplo:

HTTPPROXY=proxy.example.com:8080

En otros casos, este parámetro no es necesario.

HTTPPROXY-USERNAME = *valor*

Especifica el nombre de usuario del proxy HTTP que se va a utilizar con **HTTPPROXY-PASSWORD** si el sistema en el que se está ejecutando DFHSC2LS utiliza un servidor proxy HTTP para acceder a Internet, y el servidor proxy HTTP utiliza autenticación básica. Puede utilizar este parámetro sólo cuando especifica **HTTPPROXY**.

HTTPPROXY-PASSWORD = *valor*

Especifica la contraseña del proxy HTTP que se va a utilizar con **HTTPPROXY-USERNAME** si el sistema en el que se está ejecutando DFHSC2LS utiliza un servidor proxy HTTP para acceder a Internet, y el servidor proxy HTTP utiliza autenticación básica. Puede utilizar este parámetro sólo cuando especifica **HTTPPROXY**.

INLINE-MAXOCCURS-LIMIT = { 1 | *valor* }

Especifica si se utiliza el contenido de repetición variable en línea basado en el atributo maxOccurs del atributo XML.

El parámetro **INLINE-MAXOCCURS-LIMIT** está disponible sólo del nivel de correlación 2.1 en adelante. El *valor* de **INLINE-MAXOCCURS-LIMIT** puede ser un entero positivo en el rango de 0 a 32 767. Un valor de 0 indica que no se utiliza la correlación en línea. Un valor de 1 garantiza que los elementos opcionales se correlacionan en línea. Si el *valor* del atributo maxOccurs es mayor que el *valor* de **INLINE-MAXOCCURS-LIMIT**, se utiliza la correlación basada en contenedor; de lo contrario, se utiliza la correlación en línea.

Cuando decide si desea que las listas de repetición variable se correlacionen en línea, tenga en cuenta la longitud de un único elemento de datos recurrentes. Si tiene algunas instancias de longitud larga, es preferible la correlación basada en contenedor, si tiene muchas instancias de longitud corta, es probable que se prefiera la correlación en línea.

LANG = **COBOL**

Especifica que el lenguaje de programación de la estructura de lenguaje de alto nivel es COBOL.

LANG = **PLI-ENTERPRISE**

Especifica que el lenguaje de programación de la estructura de lenguaje de alto nivel es Enterprise PL/I.

LANG = **PLI-OTHER**

Especifica que el lenguaje de programación de la estructura de lenguaje de alto nivel es un nivel de PL/I distinto a Enterprise PL/I.

LANG = C

Especifica que el lenguaje de programación de la estructura de lenguaje de alto nivel es C.

LANG = CPP

Especifica que el lenguaje de programación de la estructura de lenguaje de alto nivel es C++.

LOGFILE = *value*

El nombre completo del archivo de z/OS UNIX en el que DFHSC2LS escribe la información de rastreo y registro de actividad. DFHSC2LS crea el archivo, pero no la estructura de directorio, si no existe.

Normalmente, no utiliza este archivo, pero puede solicitarlo la organización de servicio de IBM si encuentra problemas con DFHSC2LS.

MAPPING-LEVEL = { 1.0 | 1.1 | 1.2 | 2.0 | 2.1 | 2.2 | 3.0 | 4.0 | 4.1 }

Especifica el nivel de correlación para que el asistente lo utilice al generar las estructuras de lenguaje y enlace XML. Se recomienda utilizar el nivel de correlación más reciente que esté disponible; para DFHSC2LS, debe utilizar un nivel de correlación de 3.0 o superior.

3.0 El tipo de datos xsd:dateTime se correlaciona con el formato ASKTIME de CICS.

4.0 Utilice este nivel de correlación con una región de CICS TS 5.2 cuando dese utilizar UTF-16.

4.1 Para un soporte de matriz truncable, utilice este nivel de correlación con una región de CICS TS 5.2 que tiene aplicado APAR PI67641, o cualquier región de CICS TS 5.3 o posterior.

MAPPING-OVERRIDES = { SAME-AS-MAPPING-LEVEL | HYPHENS-AS-UNDERSCORES | NO-ARRAY-NAME-INDEXING | UNDERSCORES-AS-HYPHENS }

Especifica si el comportamiento de la correlación predeterminado se sustituye por el nivel de correlación especificado. El valor predeterminado es SAME-AS-MAPPING-LEVEL.

SAME-AS-MAPPING-LEVEL

Este parámetro genera estructuras de lenguaje en el mismo estilo que el nivel de correlación. Es el valor predeterminado.

HYPHENS-AS-UNDERSCORES

Sólo para PL/I. Este parámetro convierte cualquier guión del documento WSDL en guiones bajos en lugar del carácter X, para mejorar la legibilidad de las estructuras de lenguaje PL/I generadas. Para obtener más información, consulte el apartado XML schema to PL/I mapping.

NO-ARRAY-NAME-INDEXING

Sólo para Enterprise PL/I. Garantiza que los nombres de campo dentro de una matriz sean exclusivos sólo dentro del ámbito de la estructura de nivel superior.

UNDERSCORES-AS-HYPHENS

Esta opción está habilitada automáticamente en el nivel de correlación 4.0.

Solo para COBOL. Convierte cualquier guión bajo en el documento XML en guiones en lugar del carácter X. Esta opción mejora la legibilidad de las estructuras de lenguaje COBOL generadas. Si se produce un conflicto de nombres de campo, los campos se numeran

para garantizar que con exclusivos. Para obtener más información, consulte el apartado “Esquema XML de correlación COBOL” en la página 322.

MINIMUM-RUNTIME-LEVEL = { MINIMUM | 3.0 | 4.1 CURRENT }

Especifica el entorno de tiempo de ejecución de CICS mínimo en el que se puede desplegar el enlace XML. Si selecciona un nivel que no coincide con los demás parámetros que ha especificado, recibe un mensaje de error. Las opciones que puede seleccionar son las siguientes:

MINIMUM

El nivel de tiempo de ejecución más bajo posible de CICS se asigna automáticamente basándose en los parámetros que ha especificado.

3.0 Especifique el nivel de tiempo de ejecución 3.0 o superior si desea utilizar el asistente XML de CICS y aprovechar las correlaciones de datos avanzadas.

4.0 El archivo de enlace de servicios web generado se despliega correctamente en una región de CICS TS 5.2 o posterior. Con este nivel de ejecución, puede utilizar un nivel de correlación de 4.0 o anterior para el parámetro **MAPPING-LEVEL**. Puede utilizar cualquier parámetro opcional en este nivel.

4.1 El archivo de enlace de servicios web generado se despliega correctamente en una región de CICS V5.2 que tiene aplicado APAR PI67641, o en cualquier región de CICS V5.3 o posterior. Con este nivel de ejecución, puede utilizar un nivel de correlación de 4.1 o anterior para el parámetro **MAPPING-LEVEL**.

CURRENT

Utilice este nivel de tiempo de ejecución para desplegar el enlace XML en una región de CICS que tiene el mismo entorno de tiempo de ejecución que la región utilizada para generar el enlace XML.

OVERWRITE-OUTPUT = NO | YES

Controle si se pueden sobrescribir los **BUNDLE** de CICS en el sistema de archivos.

NO No se sustituye ningún **BUNDLE** existente. Si se encuentra un **BUNDLE** existente, DFHLS2SC emite un mensaje de error DFHPI9689E y finaliza.

YES Se sustituye cualquier **BUNDLE** existente. Si se encuentra un **BUNDLE** existente, DFHLS2SC emite un mensaje DFHPI9683W para informarle que el archivo se ha sustituido.

PDSCP = *valor*

Especifica la página de códigos que se utiliza en los miembros de un conjunto de datos particionados, donde *valor* es un número CCSID o un número de página de códigos Java. Si no especifica este parámetro, se utiliza la página de códigos de los servicios del sistema z/OS UNIX. Por ejemplo, puede especificar PDSCP=037.

PDSLIB = *valor*

Especifica el nombre del conjunto de datos particionados que contiene el lenguaje de alto nivel generado.

PDSMEM = *valor*

Especifica el prefijo de 1 a 6 caracteres que utiliza DFHSC2LS para generar el nombre del miembro del conjunto de datos particionados que contendrá las estructuras de lenguaje de alto nivel.

DFHSC2LS genera un miembro de conjunto de datos particionados para cada operación. Genera el nombre de miembro añadiendo un número de dos dígitos al prefijo.

SCHEMA = *valor*

El nombre completo del archivo de z/OS UNIX desde el que se lee el esquema XML. DFHSC2LS crea el archivo, pero no la estructura de directorio, si no existe.

Puede utilizar un esquema XML o un documento WSDL como entrada para FHSC2LS. Debe especificar este parámetro o el parámetro **WSDL**, pero no ambos, para indicar desde dónde viene la entrada.

STRUCTURE = { PDSMEM_value | *datos* }

Nombre de la estructura de datos de alto nivel en C y C++. El valor predeterminado es el valor del parámetro **PDSMEM**.

TYPES = *valor*

Define una lista de nombres locales de tipo global para habilitar. Un *valor* de ALL indica que todos los tipos globales están habilitados. De forma predeterminada, los tipos globales no están habilitados.

WIDE-COMP3 = { FULL | NO | YES }

Controla el tamaño máximo de la longitud variable decimal empaquetada en la estructura de lenguaje COBOL o PL/I generada.

FULL Para COBOL y PL/I. DFHJS2LS genera un campo decimal empaquetado que es lo suficientemente grande como para contener todos los valores válidos. El tamaño máximo es de 31 dígitos. Es el valor predeterminado.

NO Solo para COBOL. DFHJS2LS limita la longitud variable de decimal empaquetado a 10 al generar el tipo de estructura de lenguaje COBOL COMP-3. Si el tamaño del decimal empaquetado es mayor de 18, se emite un mensaje DFHPI9022W para indicar que el tipo especificado se está restringiendo a un total de 18 dígitos.

YES Solo para COBOL. DFHJS2LS soporta el tamaño máximo de 31 al generar el tipo de estructura de lenguaje COBOL COMP-3.

Nota: Las opciones NO y YES generan campos que no pueden representar todos los valores válidos; la opción FULL evita este problema. Sin embargo, la opción FULL no permite que algunos valores no válidos se representen en el campo de decimal empaquetado. Por ejemplo, si un esquema indica que hay un máximo de cinco dígitos y un máximo de dos dígitos fraccionales, la opción FULL generará un campo de decimal empaquetado que permite siete dígitos, y esto permite espacio para valores válidos como 25000 y 999,99, pero también proporciona espacio para algunos valores no válidos como 9999,99. Cuando utilice la opción FULL, tenga cuidado de no generar valores no válidos en datos de aplicación.

WSDL = *valor*

El nombre completo del documento WSDL de z/OS UNIX.

Puede utilizar un esquema XML o un documento WSDL como entrada para FHSC2LS. Debe especificar este parámetro o el parámetro **SCHEMA**, pero no ambos, para indicar desde dónde viene la entrada.

XMLCP = { LOCAL | UTF-8 | EBCDIC-CP-US }

Especifica la página de códigos que se utiliza para generar el enlace XML.

LOCAL

Este valor es el predeterminado. Especifica que el XML se genera utilizando la página de códigos local y no se genera ninguna etiqueta de codificación en el esquema XML.

UTF-8 Especifica que el XML se genera utilizando la página de códigos UTF-8. Se genera una etiqueta de codificación en el esquema XML. Si especifica esta opción, debe asegurarse de que la codificación permanece correcta al copiar el esquema XML entre diferentes plataformas.

EBCDIC-CP-US

Especifica que el XML se genera utilizando la página de códigos US EBCDIC. Se genera una etiqueta de codificación en el esquema XML.

XSDBIND = *valor*

El nombre completo del enlace XML de z/OS UNIX. DFHSC2LS crea el archivo, pero no la estructura de directorio, si no existe. La extensión de archivo es .xsdbind.

Correlación de lenguaje de alto nivel y esquema XML

Uso de los asistentes de CICS para generar correlaciones entre estructuras de lenguaje de alto nivel y esquemas XML o documentos WSDL. Los asistentes de CICS también generan esquemas XML o documentos WSDL de estructuras de datos de lenguaje de alto nivel o viceversa.

Los programas de utilidad DFHSC2LS y DFHLS2SC se conocen colectivamente como el asistente de CICS XML. Los programas de utilidad DFHWS2LS y DFHLS2WS se conocen colectivamente como el asistente de servicios web de CICS.

- DFHLS2SC y DFHLS2WS correlacionan estructuras de lenguaje de alto nivel a esquemas XML y documentos WSDL respectivamente.
- DFHSC2LS y DFHWS2LS correlacionan esquemas XML y documentos WSDL a estructuras de lenguaje de alto nivel.

Las dos correlaciones no son simétricas:

- Si procesa una estructura de datos de lenguaje con DFHLS2SC o DFHLS2WS y, a continuación, procesa el esquema XML o documento WSDL resultante con el programa de utilidad complementario (DFHSC2LS o DFHWS2LS respectivamente), no espere que la estructura de datos final sea la misma que el original. Sin embargo, la estructura de datos final es equivalente lógicamente al original.
- Si procesa un esquema XML o documento WSDL con DFHSC2LS o DFHWS2LS y, a continuación, procesa la estructura de lenguaje resultante con el programa de utilidad complementario (DFHLS2SC o DFHLS2WS respectivamente), no espere que el esquema XML en el esquema XML o documento WSDL final sea el mismo que el original.
- En algunos casos, DFHSC2LS y DFHWS2LS generan estructuras de lenguaje que no soportan DFHLS2SC y DFHLS2WS.

Debe codificar las estructuras de lenguaje procesadas por DFHLS2SC y DFHLS2WS según las reglas del lenguaje, como se implementan en los compiladores de lenguaje que soporta CICS.

Limitaciones de correlación de datos al utilizar los asistentes de CICS:

CICS soporta correlaciones de datos bidireccionales entre estructuras de lenguaje de alto nivel y esquemas XML o documentos WSDL para que se ajusten a WSDL versión 1.1 o 2.0, con ciertas limitaciones. Estas limitaciones se aplican únicamente a las herramientas DFHWS2LS y DFHSC2LS y varían según el nivel de correlación.

Limitaciones en todos los niveles de correlación

- Sólo se soportan enlaces SOAP que utilizan la codificación literal. Por lo tanto, debe establecer el atributo `use` en un valor de `literal` ; `use="encoded"` no está soportado.
- Las definiciones de tipo de datos deben codificarse utilizando el lenguaje de definición de esquemas XML (XSD). En el esquema, los tipos de datos utilizados en el mensaje SOAP deben declararse explícitamente.
- La longitud de algunas palabras clave en la descripción de servicios web es limitada. Por ejemplo, los nombres de partes, enlace y funcionamiento están limitados a 255 caracteres. En algunos casos, la longitud del nombre de operación máxima puede ser un poco más corta.
- Cualquier error SOAP definido en la descripción de servicio web se ignora. Si desea que una aplicación de proveedor de servicios envíe un mensaje de error SOAP, utilice el mandato **EXEC CICS SOAPFAULT**.
- DFHWS2LS y DFHSC2LS soportan un sólo elemento `<xsd:any>` en un ámbito particular. Por ejemplo, no se soporta el siguiente fragmento de esquema:

```
<xsd:sequence>
<xsd:any/>
<xsd:any/>
</xsd:sequence>
```

Aquí, `<xsd:any>` puede especificar `minOccurs` y `maxOccurs` si es necesario. Por ejemplo, se soporta el siguiente fragmento de esquema:

```
<xsd:sequence>
<xsd:any minOccurs="2" maxOccurs="2"/>
</xsd:sequence>
```

- No se admiten las referencias cíclicas. Por ejemplo, donde el tipo A contiene el tipo B, que, a su vez, contiene el tipo A.
- No se soporta la recurrencia en los elementos de grupo, como los elementos `<xsd:choice>`, `<xsd:sequence>`, `<xsd:group>` o `<xsd:all>`. Por ejemplo, no se soporta el siguiente fragmento de esquema:

```
<xsd:choice maxOccurs="2">
<xsd:element name="name1" type="string"/>
</xsd:choice>
```

La excepción está en el nivel de correlación 2.1 y superior, donde `maxOccurs="1"` y `minOccurs="0"` se soportan en estos elementos.

- DFHSC2LS y DFHWS2LS no soportan tipos de datos y elementos en el mensaje SOAP que se derivan de los elementos y tipos de datos declarados en el esquema XML desde el atributo `xsi:type` o desde un grupo de sustitución, excepto en el nivel de correlación 2.2 y superior si el tipo o elemento padre se define como abstracto.
- Los elementos `<xsd:sequence>` y `<xsd:group>` incluidos dentro de un elemento `<xsd:choice>` no se soportan antes del nivel de correlación 2.2. Los elementos `<xsd:choice>` y `<xsd:all>` incluidos dentro de un elemento `<xsd:choice>` no se soportan nunca.

Compatibilidad mejorada en el nivel de correlación 1.1 y superior

Cuando el nivel de correlación es 1.1 o superior, DFHWS2LS proporciona soporte para los siguientes elementos XML y tipo de elemento:

- El elemento <xsd:list>.
- El elemento <xsd:union>.
- El tipo xsd:anySimpleType.
- El elemento <xsd:attribute>. En el nivel de correlación 1.0 este elemento se ignora.

Compatibilidad mejorada en el nivel de correlación 2.1 y superior

Cuando el nivel de correlación es 2.1 o superior, DFHWS2LS soporta los siguientes elementos XML y atributos de elemento:

- El elemento <xsd:any>.
- El tipo xsd:anyType.
- Elementos abstractos. En niveles de correlación anteriores, los elementos abstractos se soportan únicamente como tipos no terminales en una jerarquía de herencia.
- Los atributos maxOccurs y minOccurs en los elementos <xsd:all>, <xsd:choice> y <xsd:sequence>, sólo cuando maxOccurs="1" y minOccurs="0".
- Los campos "FILLER" en COBOL y los campos "*" en PL/I se suprimen. Los campos no aparecen en el WSDL generado y queda un espacio adecuado en las estructuras de datos en tiempo de ejecución.

Compatibilidad mejorada en el nivel de correlación 2.2 y superior

Cuando el nivel de correlación es 2.2 o superior, DFHSC2LS y DFHWS2LS proporciona soporte mejorado para el elemento <xsd:choice>, soportando un máximo de 255 opciones en el elemento <xsd:choice>. Para obtener más información en el soporte <xsd:choice>, consulte "Soporte para <xsd:choice>" en la página 358.

En el nivel de correlación 2.2 y superior, los asistentes de CICS soportan las siguientes correlaciones XML:

- Grupos de sustitución
- Valores fijos para elementos
- Tipos de datos abstractos

Los elementos <xsd:sequence> y <xsd:group> incluidos dentro de un elemento <xsd:choice> se soportan en un nivel de correlación 2.2 y superior. Por ejemplo, se soporta el siguiente fragmento de esquema:

```
<xsd:choice>
<xsd:element name="name1" type="string"/>
<xsd:sequence/>
</xsd:choice>
```

Si el tipo o elemento padre en el mensaje SOAP se define como abstracto, DFHSC2LS y DFHWS2LS soportan elementos y tipos de datos derivados de los elementos y tipos de datos declarados en el esquema XML.

Compatibilidad mejorada en el nivel de correlación 3.0 y superior

Cuando el nivel de correlación es 3.0 o superior, los asistentes de CICS soportan las siguientes mejoras de correlación:

- DFHSC2LS y DFHWS2LS correlacionan tipos de datos `xsd:dateTime` con formato CICS ASKTIME.
- DFHLS2WS puede generar un documento WSDL y un enlace de servicio web desde una aplicación que utiliza muchos contenedores en lugar de sólo un contenedor.
- Soporte de datos truncados que se describen por medio de una estructura de datos de longitud fija. Puede establecer este comportamiento utilizando el parámetro **DATA-TRUNCATION** en los asistentes de CICS.

Compatibilidad mejorada en el nivel de correlación 4.0 y superior

Cuando el nivel de correlación es 4.0 o superior, los asistentes de CICS soportan las siguientes mejoras de correlación:

En el nivel de correlación 4.0 y superior, DFHLS2SC y DFHLS2WS soportan la cláusula COBOL OCCURS DEPENDING ON y la correlación de matrices de carácter COBOL en series XML. Puede establecer este comportamiento utilizando el parámetro **CHAR-OCCURS** en los asistentes de CICS.

- Debe especificar el parámetro **DATA-TRUNCATION=ENABLED**.
- Las cláusulas OCCURS DEPENDING ON complejas no son compatibles. Esta limitación significa que OCCURS DEPENDING ON solo es válida para el último campo de una estructura.
- CICS no soporta nombres completos (que utilizan la palabra clave 'OF') como destino de una cláusula OCCURS DEPENDING ON, por ejemplo FIELD1 OF STRUCTURE1.
- CICS no soporta la palabra clave UNBOUNDED. Debe especificar el tamaño máximo de la tabla que está esperando la aplicación.

En el nivel de correlación 4.0 y superior, los servicios web de CICS soportan la conversión de datos de aplicación que se codifican utilizando UTF-16 Unicode.

- Cuando utiliza DFHLS2WS o DFHLS2SC, puede habilitar este comportamiento usando tipos de datos específicos del lenguaje para UTF-16.
- Cuando utiliza DFHWS2LS o DFHSC2LS, puede habilitar este comportamiento estableciendo CCSID=1200.
- CICS soporta únicamente una única página de códigos Unicode, " UTF-16BE con IBM Private Use Area " (CCSID 1200).
- No se soporta la conversión de datos de aplicación que se codifican utilizando UTF-8.

Nota: DFHLS2WS y DFHLS2SC no soportan la cláusula COBOL GROUP USAGE NATIONAL.

Correlación de COBOL con un esquema XML:

Los programas de utilidad DFHLS2SC y DFHLS2WS soportan correlaciones entre estructuras de datos de COBOL y definiciones de esquema XML.

Los nombres COBOL se convierten en nombres XML según las reglas siguientes:

1. Los nombres duplicados se convierten en exclusivos añadiendo uno o varios dígitos numéricos.
Por ejemplo, dos instancias de year pasan a ser year y year1.
2. Los guiones son cambiados por caracteres con guión bajo (_). Las series de guiones contiguos son cambiados por guiones bajos contiguos.
Por ejemplo, current-user--id se convierte en current_user__id.
3. Los segmentos de nombres que están delimitados por guiones y que sólo contienen caracteres en mayúsculas se convierten en minúsculas.
Por ejemplo, CA-REQUEST-ID pasa a ser ca_request_id.
4. Se añade un guión bajo inicial a los nombres que empiezan por un carácter numérico.
Por ejemplo, 9A-REQUEST-ID pasa a ser _9a_request_id.

CICS correlaciona elementos de descripción de datos COBOL con elementos de esquema de acuerdo con la tabla siguiente. Los elementos de descripción de datos de COBOL que no se muestran en la tabla no son compatibles con DFHLS2SC o DFHLS2WS. También se aplican las siguientes restricciones:

- Los elementos de descripción de datos con números de nivel de 66 y 77 no se soportan. Los elementos de descripción de datos con un número de nivel de 88 se ignoran.
- No se soportan las siguientes cláusulas en entradas de descripción de datos:
REDEFINES
RENAMES; es el nivel 66
DATE FORMAT
- Se ignoran las siguientes cláusulas en los elementos de descripción de datos:
BLANK WHEN ZERO
JUSTIFIED
VALUE
- Se soporta la cláusula SIGN "SIGN TRAILING". SIGN LEADING de la cláusula SIGN solo se admite cuando el nivel de correlación especificado en DFHLS2SC o DFHLS2WS es 1.2 o superior.
- Se soporta SEPARATE CHARACTER en un nivel de correlación de 1.2 o superior tanto para la cláusula SIGN TRAILING como para SIGN LEADING.
- No se soportan las siguientes frases en la cláusula USAGE:
OBJECT REFERENCE
POINTER
FUNCTION-POINTER
PROCEDURE-POINTER
- Se soportan las siguientes frases en la cláusula USAGE en un nivel de correlación de 1.2 o superior:
COMPUTATIONAL-1
COMPUTATIONAL-2
- Los únicos caracteres PICTURE que se admiten para los elementos de descripción de datos DISPLAY y COMPUTATIONAL-5 son 9, S y Z.
- Los caracteres PICTURE que se admiten para los elementos de descripción de datos PACKED-DECIMAL son 9, S, V y Z.
- Los únicos caracteres PICTURE que se admiten con los elementos de descripción de datos numéricos editados son 9 y Z.

- Si el parámetro **MAPPING-LEVEL** se establece en 1.2 o superior y el parámetro **CHAR-VARYING** se establece en NULL, las matrices de caracteres se correlacionan con una `xsd:string` y se procesan como series terminadas en nulo.
- Si el parámetro **MAPPING-LEVEL** se establece en 1.2 o superior y el parámetro **CHAR-VARYING** se establece en BINARY, las correlaciones de caracteres se correlacionan con `xsd:base64Binary` y se procesan como datos binarios.
- Si el parámetro **MAPPING-LEVEL** está establecido en 1.2 o superior y el parámetro **CHAR-VARYING** está establecido en COLLAPSE, se ignora el espacio en blanco final para series.
- La cláusula OCCURS DEPENDING ON es compatible a nivel de correlación 4.0 o superior. Las cláusulas OCCURS DEPENDING ON complejas no son compatibles. Esto significa que OCCURS DEPENDING ON sólo se soporta para el último campo de una estructura.
- La cláusula OCCURS INDEXED BY es compatible a nivel de correlación.
- Se admiten hasta 65535 apariciones de la cláusula OCCURS. Esto significa OCCURS *n* TIMES donde *n* es mayor que 65535 y no está soportado.

Descripción de datos COBOL	Esquema simpleType
<div> <div><i>n</i></div> <div>PIC X(</div> <div>)</div> </div> <div> <div><i>n</i></div> <div>PIC A(</div> <div>)</div> </div> <div> <div><i>n</i></div> <div>PIC G(</div> <div>) DISPLAY-1</div> </div> <div> <div><i>n</i></div> <div>PIC N(</div> <div>)</div> </div>	<pre> <xsd:simpleType> <xsd:restriction base="xsd:string"> <xsd:maxLength value=" <i>n</i>" /> <xsd:whiteSpace value="preserve" /> </xsd:restriction> </xsd:simpleType> </pre>
<div>PIC S9 DISPLAY</div> <div>PIC S99 DISPLAY</div> <div>PIC S999 DISPLAY</div> <div>PIC S9999 DISPLAY</div>	<pre> <xsd:simpleType> <xsd:restriction base="xsd:short"> <xsd:minInclusive value="- <i>n</i>" /> <xsd:maxInclusive value=" <i>n</i>" /> </xsd:restriction> </xsd:simpleType> </pre> <p>donde <i>n</i> es el valor máximo que puede ser representado por el patrón de '9' caracteres.</p>
<div> <div><i>z</i></div> <div>PIC S9(</div> <div>) DISPLAY</div> </div> <div>donde $5 \leq z \leq 9$</div>	<pre> <xsd:simpleType> <xsd:restriction base="xsd:int"> <xsd:minInclusive value="- <i>n</i>" /> <xsd:maxInclusive value=" <i>n</i>" /> </xsd:restriction> </xsd:simpleType> </pre> <p>donde <i>n</i> es el valor máximo que puede ser representado por el patrón de '9' caracteres.</p>

Descripción de datos COBOL	Esquema simpleType
PIC S9(z) DISPLAY donde $9 < z$	<pre><xsd:simpleType> <xsd:restriction base="xsd:long"> <xsd:minInclusive value="- n"/> <xsd:maxInclusive value=" n"/> </xsd:restriction> </xsd:simpleType></pre> <p>donde n es el valor máximo que puede ser representado por el patrón de '9' caracteres.</p>
PIC 9 DISPLAY PIC 99 DISPLAY PIC 999 DISPLAY PIC 9999 DISPLAY	<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedShort"> <xsd:minInclusive value="0"/> <xsd:maxInclusive value=" n"/> </xsd:restriction> </xsd:simpleType></pre> <p>donde n es el valor máximo que puede ser representado por el patrón de '9' caracteres.</p>
PIC 9(z) DISPLAY donde $5 \leq z \leq 9$	<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedInt"> <xsd:minInclusive value="0"/> <xsd:maxInclusive value=" n"/> </xsd:restriction> </xsd:simpleType></pre> <p>donde n es el valor máximo que puede ser representado por el patrón de '9' caracteres.</p>
PIC 9(z) DISPLAY donde $9 < z$	<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedLong"> <xsd:minInclusive value="0"/> <xsd:maxInclusive value=" n"/> </xsd:restriction> </xsd:simpleType></pre> <p>donde n es el valor máximo que puede ser representado por el patrón de '9' caracteres.</p>
PIC S9(n) COMP PIC S9(n) COMP-4 PIC S9(n) COMP-5 PIC S9(n) BINARY donde $n \leq 4$.	<pre><xsd:simpleType> <xsd:restriction base="xsd:short"> </xsd:restriction> </xsd:simpleType></pre>

Descripción de datos COBOL	Esquema simpleType
<div>PIC S9(n) COMP</div> <div>PIC S9(n) COMP-4</div> <div>PIC S9(n) COMP-5</div> <div>PIC S9(n) BINARY</div> <p>donde $5 \leq n \leq 9$.</p>	<pre><xsd:simpleType> <xsd:restriction base="xsd:int"> </xsd:restriction> </xsd:simpleType></pre>
<div>PIC S9(n) COMP</div> <div>PIC S9(n) COMP-4</div> <div>PIC S9(n) COMP-5</div> <div>PIC S9(n) BINARY</div> <p>donde $9 < n$.</p>	<pre><xsd:simpleType> <xsd:restriction base="xsd:long"> </xsd:restriction> </xsd:simpleType></pre>
<div>PIC 9(n) COMP</div> <div>PIC 9(n) COMP-4</div> <div>PIC 9(n) COMP-5</div> <div>PIC 9(n) BINARY</div> <p>donde $n \leq 4$.</p>	<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedShort"> </xsd:restriction> </xsd:simpleType></pre>

Descripción de datos COBOL	Esquema simpleType
PIC 9(n) COMP PIC 9(n) COMP-4 PIC 9(n) COMP-5 PIC 9(n) BINARY donde $5 \leq n \leq 9$.	<pre> <xsd:simpleType> <xsd:restriction base="xsd:unsignedInt"> </xsd:restriction> </xsd:simpleType> </pre>
PIC 9(n) COMP PIC 9(n) COMP-4 PIC 9(n) COMP-5 PIC 9(n) BINARY donde $9 < n$.	<pre> <xsd:simpleType> <xsd:restriction base="xsd:unsignedLong"> </xsd:restriction> </xsd:simpleType> </pre>
PIC S9(m)V9(n) COMP-3 donde $p = m + n$.	<pre> <xsd:simpleType> <xsd:restriction base="xsd:decimal"> <xsd:totalDigits value=" p" /> <xsd:fractionDigits value=" n" /> </xsd:restriction> </xsd:simpleType> </pre> <p>donde $p = m + n$.</p>

Descripción de datos COBOL	Esquema simpleType
<p>PIC 9(<i>m</i>)V9(<i>n</i>) COMP-3</p> <p>PIC S9(<i>m</i>) COMP-3</p> <p>Se soporta en el nivel de correlación 3.0 cuando DATETIME=PACKED15</p>	<pre> <xsd:simpleType> <xsd:restriction base="xsd:decimal"> <xsd:totalDigits value=" <i>p</i>"/> <xsd:fractionDigits value=" <i>n</i>"/> <xsd:minInclusive value="0"/> </xsd:restriction> </xsd:simpleType> </pre> <p>donde $p = m + n$.</p> <pre> <xsd:simpleType> <xsd:restriction base="xsd:dateTime"> </xsd:restriction> </xsd:simpleType> </pre> <p>El formato de la indicación de fecha y hora es CICS ABSTIME.</p>
<p>PIC S9(<i>m</i>)V9(<i>n</i>) DISPLAY</p> <p>Se soporta en el nivel de correlación 1.2 y superior</p>	<pre> <xsd:simpleType> <xsd:restriction base="xsd:decimal"> <xsd:totalDigits value=" <i>p</i>"/> <xsd:fractionDigits value=" <i>n</i>"/> </xsd:restriction> </xsd:simpleType> </pre> <p>donde $p = m + n$.</p>
<p>COMP-1</p> <p>Se soporta en el nivel de correlación 1.2 y superior</p> <p>Nota: La representación de datos de coma flotante hexadecimal (HFP) de IBM no es exactamente lo mismo que la representación IEEE-754-1985 que se utiliza para XML. Algunos valores podrían no convertirse de forma exacta de una representación a la otra. Algunos valores sumamente grandes o pequeños podrían no ser válidos para los tipos de datos xsd:float. Algunos valores podrían perder precisión cuando se convierten a o desde la representación HFP. Si las conversiones precisas son importantes, considere sustituir el uso de tipos de datos COMP-1 por alternativas de precisión fija.</p>	<pre> <xsd:simpleType> <xsd:restriction base="xsd:float"> </xsd:restriction> </xsd:simpleType> </pre>

Descripción de datos COBOL	Esquema simpleType
<p>COMP-2</p> <p>Se soporta en el nivel de correlación 1.2 y superior</p> <p>Nota: La representación de datos de coma flotante hexadecimal (HFP) de IBM no es exactamente lo mismo que la representación IEEE-754-1985 que se utiliza para XML. Algunos valores podrían no convertirse de forma exacta de una representación a la otra. Algunos valores sumamente grandes o pequeños podrían no ser válidos para los tipos de datos xsd:double. Algunos valores podrían perder precisión cuando se convierten a o desde la representación HFP. Si las conversiones precisas son importantes, considere sustituir el uso de tipos de datos COMP-2 por alternativas de precisión fija.</p>	<pre><xsd:simpleType> <xsd:restriction base="xsd:double"> </xsd:restriction> </xsd:simpleType></pre>
<p><i>descripción de datos OCCURS n TIMES</i></p>	<pre><xsd:element name= "field-name" minOccurs= "n" maxOccurs= "n"> ... </xsd:element></pre> <p>El contenido del elemento depende del tipo de datos utilizado.</p>
<p><i>de datos</i></p> <p><i>descripción</i></p> <p>OCCURS</p> <p><i>n</i></p> <p>TO</p> <p><i>m</i></p> <p>TIMES</p> <p>DEPENDING ON</p> <p><i>t</i></p> <p>Se admite en el nivel de correlación 4.0</p>	<pre><xsd:element name= "field-name" minOccurs= "n" maxOccurs= "m"> ... </xsd:element></pre>

Descripción de datos COBOL	Esquema simpleType
PIC X OCCURS <i>n</i> TIMES PIC A OCCURS <i>n</i> TIMES PIC G DISPLAY-1 OCCURS <i>n</i> TIMES PIC N OCCURS <i>n</i> TIMES	<p>Cuando CHAR-OCCURS =STRING :</p> <pre> <xsd:element name= "field-name" > <xsd:simpleType> <xsd:restriction base= "s:string"> <xsd:maxLength value= "n"> </xsd:restriction> </xsd:simpleType> </xsd:element> </pre> <p>Esta es una serie.</p> <p>Cuando CHAR-OCCURS =ARRAY :</p> <pre> <xsd:element name= "field-name" minOccurs= "n" maxOccurs= "n"> <xsd:simpleType> <xsd:restriction base= "s:string"> <xsd:maxLength value= "1"> </xsd:restriction> </xsd:simpleType> </xsd:element> </pre> <p>Es una matriz de caracteres individuales.</p>

Descripción de datos COBOL	Esquema simpleType
PIC X OCCURS <i>n</i> TO <i>m</i> TIMES DEPENDING ON <i>t</i> PIC A OCCURS <i>n</i> TO <i>m</i> TIMES DEPENDING ON <i>t</i> PIC G DISPLAY-1 OCCURS <i>n</i> TO <i>m</i> TIMES DEPENDING ON <i>t</i> PIC N OCCURS <i>n</i> TO <i>m</i> TIMES DEPENDING ON <i>t</i>	Cuando CHAR-OCCURS =STRING : <xsd:element name= "field-name" > <xsd:simpleType> <xsd:restriction base= "s:string"> <xsd:maxLength value= "m"> <xsd:minLength value= "n"> </xsd:restriction> </xsd:simpleType> </xsd:element>
PIC N(<i>n</i>) USAGE NATIONAL Cuando CHAR-USAGE =NATIONAL : PIC N(<i>n</i>)	<xsd:simpleType> <xsd:restriction base="xsd:string"> <xsd:maxLength value="n"/> <xsd:whiteSpace value="preserve"/> </xsd:restriction> </xsd:simpleType> En tiempo de ejecución, CICS rellena el campo de estructura de datos de aplicación con datos UTF-16.

Esquema XML de correlación COBOL:

Los programas de utilidad DFHSC2LS y DFHWS2LS soportan la correlación entre definiciones de esquema XML y estructuras de datos COBOL.

Los asistentes de CICS generan nombres exclusivos y válidos para variables COBOL de los nombres de elemento del esquema utilizando las reglas siguientes:

1. Las palabras reservadas de COBOL llevan el prefijo ' X '.
Por ejemplo, DISPLAY pasa a ser XDISPLAY.
2. Caracteres que no sean A-Z, a-z, 0-9 o guión se sustituyen por ' X '.
Por ejemplo, monthly_total pasa a ser monthlyXtotal. Puede utilizar el parámetro **MAPPING-OVERRIDES** para cambiar la forma en la que se manejan otros

caracteres. Por ejemplo, si establece el valor `UNDERSCORES-AS-HYPHENS`, cualquier guión bajo de XML se convierte en un guión en lugar de en una X. Por ejemplo, `monthly_total` pasa a ser `monthly-total`.

3. Si el último carácter es un guión se sustituye por ' X '.

Por ejemplo, `ca-request-` pasa a ser `ca-requestX`.

4. Si el esquema especifica que la variable tiene cardinalidad variable (es decir, se especifican `minOccurs` y `maxOccurs` en un `xsd:element` con valores diferentes), y el nombre de elemento de esquema es mayor de 23 caracteres, se trunca en esa longitud.

Si el esquema especifica que la variable tiene cardinalidad fija y el nombre de elemento de esquema es mayor de 28 caracteres, se trunca en esa longitud.

5. Los nombres duplicados en el mismo ámbito se convierten en exclusivos añadiendo uno o dos dígitos numéricos a la segunda y siguientes instancias del nombre.

Por ejemplo, tres instancias de `year` pasan a ser `year`, `year1` y `year2`.

6. Se reservan cinco caracteres para las series `-cont` o `-num`, que se utilizan cuando el esquema especifica que la variable tiene una cardinalidad variable; es decir, cuando `minOccurs` y `maxOccurs` se especifican con valores diferentes.

Para obtener más información, consulte el apartado "Matrices de variables de elementos" en la página 349.

7. Para los atributos, se aplican las reglas anteriores al elemento. Se añade el prefijo `attr-` al nombre de elemento y va seguido de `-value` o `-exist`. Si la longitud total es mayor de 28 caracteres, el nombre de elemento se trunca. Para obtener más información, consulte el apartado "Soporte para atributos XML" en la página 354.

El atributo `nillable` tiene reglas especiales. Se añade el prefijo `attr-`, pero también se añade `nil-` al principio del nombre de elemento. El nombre de elemento va seguido de `-value`. Si la longitud total es mayor de 28 caracteres, el nombre de elemento se trunca.

La longitud total del nombre resultante es de 30 caracteres o menos.

DFHSC2LS y DFHWS2LS correlacionan tipos de esquema con elementos de descripción de datos COBOL utilizando el nivel de correlación especificado según la tabla siguiente. Tenga en cuenta los puntos siguientes:

- Si el parámetro **MAPPING-LEVEL** se establece en 1.2 o superior y el parámetro **CHAR-VARYING** se establece en `NULL`, los datos de caracteres de longitud variable se correlacionan con series terminadas en nulo y se asigna un carácter adicional al terminador en nulo.
- Si el parámetro **MAPPING-LEVEL** se establece en 1.2 o superior y el parámetro **CHAR-VARYING** se establece en `YES`, los datos de carácter de longitud variable se correlacionan con dos elementos relacionados: un campo de longitud y un campo de datos. Por ejemplo:

```
<xsd:simpleType name="VariableStringType">
<xsd:restriction base="xsd:string">
<xsd:minLength value="1"/>
<xsd:maxLength value="10000"/>
</xsd:restriction>
</xsd:simpleType>
<xsd:element name="textString" type="tns:VariableStringType"/>
```

se correlaciona con:

```
15 textString-length PIC S9999 COMP-5 SYNC
15 textString PIC X(10000)
```

Tipo simple de esquema	Descripción de datos COBOL
<pre><xsd:simpleType> <xsd:restriction base="xsd:anyType"> </xsd:restriction> </xsd:simpleType></pre>	<p>Nivel de correlación 2.0 e inferior:</p> <p>No soportada</p> <p>Nivel de correlación 2.1:</p> <p>Soportada</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:anySimpletype"> </xsd:restriction> </xsd:simpleType></pre>	<p>Nivel de correlación 1.0:</p> <p>No soportada</p> <p>Nivel de correlación 1.1 y superior:</p> <p>PIC X(255)</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd: type" <xsd:length value=" z"/> </xsd:restriction> </xsd:simpleType></pre> <p>donde <i>type</i> es uno de:</p> <ul style="list-style-type: none"> string normalizedString token Name NMTOKEN language NCName ID IDREF ENTITY hexBinary 	<p>Todos los niveles de correlación:</p> <p>PIC X(</p> <p>z</p> <p>)</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd: type"> <xsd:length value=" z"/> </xsd:restriction> </xsd:simpleType></pre> <p>donde <i>type</i> es uno de:</p> <ul style="list-style-type: none"> string normalizedString token Name NMTOKEN language NCName ID IDREF ENTITY 	<p>Cuando CCSID=1200 está en el nivel de correlación 4.0 y superior:</p> <p>PIC N(</p> <p>z</p> <p>) USAGE NATIONAL</p>

Tipo simple de esquema	Descripción de datos COBOL
<pre><xsd:simpleType> <xsd:restriction base="xsd: type" </xsd:restriction> </xsd:simpleType></pre> <p>donde <i>type</i> es uno de:</p> <ul style="list-style-type: none"> duration date time gDay gMonth gYear gMonthDay gYearMonth 	<p>Todos los niveles de correlación:</p> <p>PIC X(32)</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:dateTime" </xsd:restriction> </xsd:simpleType></pre>	<p>Nivel de correlación 1.2 e inferior:</p> <p>PIC X(32)</p> <p>Nivel de correlación 2.0 y superior:</p> <p>PIC X(40)</p> <p>Nivel de correlación 3.0 y superior:</p> <p>PIC S9(15) COMP-3</p> <p>El formato es CICS ABSTIME.</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd: type"> </xsd:restriction> </xsd:simpleType></pre> <p>donde <i>type</i> es uno de:</p> <ul style="list-style-type: none"> byte unsignedByte 	<p>Todos los niveles de correlación:</p> <p>PIC X DISPLAY</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:short"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos los niveles de correlación:</p> <p>PIC S9999 COMP-5 SYNC</p> <p>o</p> <p>PIC S9999 DISPLAY</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedShort"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos los niveles de correlación:</p> <p>PIC 9999 COMP-5 SYNC</p> <p>o</p> <p>PIC 9999 DISPLAY</p>

Tipo simple de esquema	Descripción de datos COBOL
<pre><xsd:simpleType> <xsd:restriction base="xsd:integer"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos los niveles de correlación:</p> <p>PIC S9(18) COMP-3</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:int"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos los niveles de correlación:</p> <p>PIC S9(9) COMP-5 SYNC</p> <p>o</p> <p>PIC S9(9) DISPLAY</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedInt"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos los niveles de correlación:</p> <p>PIC 9(9) COMP-5 SYNC</p> <p>o</p> <p>PIC 9(9) DISPLAY</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:long"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos los niveles de correlación:</p> <p>PIC S9(18) COMP-5 SYNC</p> <p>o</p> <p>PIC S9(18) DISPLAY</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedLong"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos los niveles de correlación:</p> <p>PIC 9(18) COMP-5 SYNC</p> <p>o</p> <p>PIC 9(18) DISPLAY</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:decimal"> <xsd:totalDigits value=" m" <xsd:fractionDigits value=" n" </xsd:restriction> </xsd:simpleType></pre>	<p>Todos los niveles de correlación:</p> <p>Si WIDE-COMP3=FULL:</p> <p>PIC 9(</p> <p>m</p> <p>)V9(</p> <p>n</p> <p>) COMP-3</p> <p>De lo contrario:</p> <p>PIC 9(</p> <p>p</p> <p>)V9(</p> <p>n</p> <p>) COMP-3</p> <p>donde $p = m - n$.</p>

Tipo simple de esquema	Descripción de datos COBOL
<pre><xsd:simpleType> <xsd:restriction base="xsd:boolean"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos los niveles de correlación:</p> <p>PIC X DISPLAY</p> <p>El valor x'00' implica falso, x'01' implica verdadero.</p>
<pre><xsd:simpleType> <xsd:list> <xsd:simpleType> <xsd:restriction base="xsd:int"/> </xsd:simpleType> </xsd:list> </xsd:simpleType></pre>	<p>Nivel de correlación 1.0:</p> <p>No soportada</p> <p>Nivel de correlación 1.1 y superior:</p> <p>PIC X(255)</p>
<pre><xsd:simpleType> <xsd:union memberTypes="xsd:int xsd:string"/> </xsd:simpleType></pre>	<p>Nivel de correlación 1.0:</p> <p>No soportada</p> <p>Nivel de correlación 1.1 y superior:</p> <p>PIC X(255)</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:base64Binary"> <xsd:length value=" z"/> </xsd:restriction> </xsd:simpleType> <xsd:simpleType> <xsd:restriction base="xsd:base64Binary"> </xsd:restriction> </xsd:simpleType></pre> <p>donde la longitud no está definida.</p>	<p>Nivel de correlación 1.0:</p> <p>No soportado</p> <p>Nivel de correlación 1.1:</p> <p>PIC X(</p> <p>y</p> <p>)</p> <p>donde $y = 4 \times (\text{ceil}(z / 3))$. $\text{ceil}(x)$ es el entero más pequeño mayor o igual que x.</p> <p>Nivel de correlación 1.2 y superior:</p> <p>PIC X(</p> <p>z</p> <p>)</p> <p>donde la longitud es fija.</p> <p>PIC X(16)</p> <p>donde la longitud no está definida. El campo contiene el nombre de 16 bytes del contenedor que almacena los datos binarios.</p>

Tipo simple de esquema	Descripción de datos COBOL
<pre><xsd:simpleType> <xsd:restriction base="xsd:float"> </xsd:restriction> </xsd:simpleType></pre>	<p>Nivel de correlación 1.1 e inferior:</p> <p>PIC X(32)</p> <p>Nivel de correlación 1.2 y superior:</p> <p>COMP-1</p> <p>Nota: La representación de datos de coma flotante hexadecimal (HFP) de IBM no es exactamente lo mismo que la representación IEEE-754-1985 que se utiliza para XML. Algunos valores podrían no convertirse de forma exacta de una representación a la otra. Algunos valores sumamente grandes o pequeños podrían no ser válidos para los tipos de datos <code>xsd:float</code>. Algunos valores podrían perder precisión cuando se convierten a o desde la representación HFP. Si las conversiones precisas son importantes, considere sustituir el uso de tipos de datos COMP-1 por alternativas de precisión fija.</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:double"> </xsd:restriction> </xsd:simpleType></pre>	<p>Nivel de correlación 1.1 e inferior:</p> <p>PIC X(32)</p> <p>Nivel de correlación 1.2 y superior:</p> <p>COMP-2</p> <p>Nota: La representación de datos de coma flotante hexadecimal (HFP) de IBM no es exactamente lo mismo que la representación IEEE-754-1985 que se utiliza para XML. Algunos valores podrían no convertirse de forma exacta de una representación a la otra. Algunos valores sumamente grandes o pequeños podrían no ser válidos para los tipos de datos <code>xsd:double</code>. Algunos valores podrían perder precisión cuando se convierten a o desde la representación HFP. Si las conversiones precisas son importantes, considere sustituir el uso de tipos de datos COMP-2 por alternativas de precisión fija.</p>

Algunos de los tipos de esquema que se muestran en la tabla se correlacionan con un formato COBOL de COMP-5 SYNC o de DISPLAY, dependiendo de los valores (si los hubiera) que se especifican en las facetas `minInclusive` y `maxInclusive`:

- Para tipos firmados (`short` , `int` y `long`), se utiliza DISPLAY cuando se especifica lo siguiente:

```
<xsd:minInclusive value="-
a"/>
<xsd:maxInclusive value="
a
"/>
```

donde *a* es una serie de '9'.

- Para los tipos sin firmar (`unsignedShort` , `unsignedInt` y `unsignedLong`), se utiliza DISPLAY cuando se especifica lo siguiente:


```

<xsd:minInclusive value="0"/>
<xsd:maxInclusive value="
a
"/>

```

donde *a* es una serie de '9'.

Cuando se especifica otro valor, o no se especifica ningún valor, se utiliza COMP-5 SYNC.

Correlación de C y C++ con un esquema XML:

Los programas de utilidad DFHLS2SC y DFHLS2WS soportan correlaciones entre los tipos de datos C y C++ y las definiciones de esquema XML.

Los nombres C y C++ se convierten en nombres XML según las reglas siguientes:

1. Los caracteres que no son válidos en los nombres de elemento XML se sustituyen por 'X'.
Por ejemplo, `monthly-total` pasa a ser `monthlyXtotal`.
2. Los nombres duplicados se convierten en exclusivos añadiendo uno o varios dígitos numéricos.
Por ejemplo, dos instancias de `year` pasan a ser `year` y `year1`.

DFHLS2SC y DFHLS2WS correlacionan tipos de datos C y C++ con elementos de esquema según la tabla siguiente. Los tipos C y C++ que no se muestran en la tabla no son compatibles con DFHLS2SC o DFHLS2WS. El calificador `_Packed` se soporta para las estructuras. Se aplican estas restricciones:

- Los archivos de cabecera deben contener una instancia de alto nivel `struct`.
- No puede declarar un tipo de estructura que se contiene a sí mismo como miembro.
- Los siguientes tipos de datos C y C++ no se soportan:
`decimal`
`long double`
`wchar_t` (sólo C++)
- Los siguientes caracteres se ignoran si están presentes en el archivo de cabecera.

Especificadores de clase de almacenamiento:

```

auto
register
static
extern
mutable

```

Calificadores

```

const
volatile
_Export (sólo C++)

```

Especificadores de función

```

inline (sólo C++)
Virtual (sólo C++)

```

Valores iniciales

- El archivo de cabecera no debe contener estos elementos:
`Uniones`
`Declaraciones de clase`
`Tipos de datos de enumeración`

Variables de tipo de puntero

Declaraciones de plantilla

Las macros predefinidas; es decir, macros son nombres que comienzan y finalizan con dos caracteres de subrayado (__)

La secuencia de continuación de línea (un símbolo \ que va seguido inmediatamente por un carácter de nueva línea)

Declaradores de funciones de prototipos

Directivas de preprocesamiento

Campos de bits

La palabra clave __cdecl (o _cdecl) (sólo C++)

- El programador de aplicación debe utilizar un compilador de 32 bits para asegurar que un int se correlaciona con 4 bytes.
- No se soportan las siguientes palabras clave de C++ reservadas:
 - explicit
 - using
 - namespace
 - typename
 - typeid
- Si el parámetro **MAPPING-LEVEL** se establece en 1.2 o superior y el parámetro **CHAR-VARYING** se establece en NULL, las matrices de caracteres se correlacionan con una xsd:string y se procesan como series terminadas en nulo.
- Si el parámetro **MAPPING-LEVEL** se establece en 1.2 o superior y el parámetro **CHAR-VARYING** se establece en BINARY, las correlaciones de caracteres se correlacionan con xsd:base64Binary y se procesan como datos binarios.
- Si el parámetro **MAPPING-LEVEL** se establece en 1.2 o superior y el parámetro **CHAR-VARYING** se establece en COLLAPSE, <xsd:whiteSpace value="collapse"/> se genera para las series.

Tipos de datos C y C++	Esquema simpleType
char[z]	<pre><xsd:simpleType> <xsd:restriction base="xsd:string"> <xsd:length value=" z"/> </xsd:restriction> </xsd:simpleType></pre>
char16_t[n]	<p>En el nivel de correlación 4.0 y superior:</p> <pre><xsd:simpleType> <xsd:restriction base="xsd:string"> <xsd:maxlength value="n"/> <xsd:whiteSpace value="preserve"/> </xsd:restriction> </xsd:simpleType></pre> <p>En tiempo de ejecución, CICS rellena el campo de estructura de datos de aplicación con datos UTF-16.</p>
char[8] Se soporta en el nivel de correlación 3.0 y superior cuando DATETIME=PACKED15	<pre><xsd:simpleType> <xsd:restriction base="xsd:dateTime" </xsd:restriction> </xsd:simpleType></pre> <p>El formato de la indicación de fecha y hora es CICS ABSTIME.</p>

Tipos de datos C y C++	Esquema simpleType
char	<pre><xsd:simpleType> <xsd:restriction base="xsd:byte"> </xsd:restriction> </xsd:simpleType></pre>
unsigned char	<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedByte"> </xsd:restriction> </xsd:simpleType></pre>
short	<pre><xsd:simpleType> <xsd:restriction base="xsd:short"> </xsd:restriction> </xsd:simpleType></pre>
unsigned short	<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedShort"> </xsd:restriction> </xsd:simpleType></pre>
int long	<pre><xsd:simpleType> <xsd:restriction base="xsd:int"> </xsd:restriction> </xsd:simpleType></pre>
unsigned int unsigned long	<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedInt"> </xsd:restriction> </xsd:simpleType></pre>
long long	<pre><xsd:simpleType> <xsd:restriction base="xsd:long"> </xsd:restriction> </xsd:simpleType></pre>
unsigned long long	<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedLong"> </xsd:restriction> </xsd:simpleType></pre>
bool (sólo C++)	<pre><xsd:simpleType> <xsd:restriction base="xsd:boolean"> </xsd:restriction> </xsd:simpleType></pre>
float Se soporta en el nivel de correlación 1.2 y superior	<pre><xsd:simpleType> <xsd:restriction base="xsd:float"> </xsd:restriction> </xsd:simpleType></pre> <p>Nota: La representación de datos de coma flotante hexadecimal (HFP) de IBM no es exactamente lo mismo que la representación IEEE-754-1985 que se utiliza para XML. Algunos valores podrían no convertirse de forma exacta de una representación a la otra. Algunos valores sumamente grandes o pequeños podrían no ser válidos para los tipos de datos xsd:float. Algunos valores podrían perder precisión cuando se convierten a o desde la representación HFP. Si las conversiones precisas son importantes, considere sustituir el uso de tipos de datos flotantes por alternativas de precisión fija.</p>

Tipos de datos C y C++	Esquema simpleType
double Se soporta en el nivel de correlación 1.2 y superior	<pre><xsd:simpleType> <xsd:restriction base="xsd:double"> </xsd:restriction> </xsd:simpleType></pre> <p>Nota: La representación de datos de coma flotante hexadecimal (HFP) de IBM no es exactamente lo mismo que la representación IEEE-754-1985 que se utiliza para XML. Algunos valores podrían no convertirse de forma exacta de una representación a la otra. Algunos valores sumamente grandes o pequeños podrían no ser válidos para los tipos de datos xsd:double. Algunos valores podrían perder precisión cuando se convierten a o desde la representación HFP. Si las conversiones precisas son importantes, considere sustituir el uso de tipos de datos dobles por alternativas de precisión fija.</p>

Correlación de esquema XML con C y C++:

Los programas de utilidad DFHSC2LS y DFHWS2LS soportan correlaciones entre las definiciones de esquema XML que se incluyen en cada una de las descripciones de servicio web y los tipos de datos C y C++.

Los asistentes de CICS generan nombres exclusivos y válidos para variables C y C++ de los nombres de elemento de esquema que utilizan las reglas siguientes:

1. Caracteres que no sean A-Z, a-z, 0-9, o _ se sustituyen por ' X '.
Por ejemplo, monthly-total pasa a ser monthlyXtotal.
2. Si el primer carácter no es un carácter alfabético, se sustituye por una ' X '.
Por ejemplo, _monthlysummary pasa a ser Xmonthlysummary.
3. Si el nombre de elemento de esquema es mayor de 50 caracteres, se trunca en esa longitud.
4. Los nombres duplicados en el mismo ámbito se convierten en exclusivos añadiendo uno o varios dígitos numéricos.
Por ejemplo, dos instancias de year pasan a ser year y year1.
5. Se reservan cinco caracteres para las series _cont o _num , que se utilizan cuando el esquema especifica que la variable tiene una cardinalidad variable; es decir, cuando minOccurs y maxOccurs se especifican en xsd:element.
Para obtener más información, consulte el apartado “Matrices de variables de elementos” en la página 349.
6. Para los atributos, se aplican las reglas anteriores al elemento. El prefijo attr_ se añade al nombre de elemento y va seguido de _value o _exist. Si la longitud total es mayor de 28 caracteres, el nombre de elemento se trunca.
El atributo nillable tiene reglas especiales. Se añade el prefijo attr_ , pero también se añade nil_ al principio del nombre de elemento. El nombre de elemento va seguido de _value. Si la longitud total es mayor de 28 caracteres, el nombre de elemento se trunca.

La longitud total del nombre resultante es de 57 caracteres o menos.

DFHSC2LS y DFHWS2LS correlacionan tipos de esquema para C y C++ según la tabla siguiente. También se aplican las reglas siguientes:

- Si el parámetro **MAPPING-LEVEL** se establece en 1.2 o superior y el parámetro **CHAR-VARYING** se establece en NULL, los datos de caracteres de longitud variable se correlacionan con series terminadas en nulo y se asigna un carácter adicional al terminador en nulo.

- Si el parámetro **MAPPING-LEVEL** se establece en 1.2 o superior y el parámetro **CHAR-VARYING** se establece en YES, los datos de carácter de longitud variable se correlacionan con dos elementos relacionados: un campo de longitud y un campo de datos.

Esquema simpleType	Tipos de datos C y C++
<pre><xsd:simpleType> <xsd:restriction base="xsd:anyType"> </xsd:restriction> </xsd:simpleType></pre>	<p>Nivel de correlación 2.0 e inferior:</p> <p>No soportado</p> <p>Nivel de correlación 2.1 y superior:</p> <p>Soportado</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:anySimpletype"> </xsd:restriction> </xsd:simpleType></pre>	<p>Nivel de correlación 1.0:</p> <p>No soportado</p> <p>Nivel de correlación 1.1 y superior:</p> <p>char[255]</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd: type"> <xsd:length value=" z"/> </xsd:restriction> </xsd:simpleType></pre> <p>donde <i>type</i> es uno de:</p> <ul style="list-style-type: none"> string normalizedString token Name NMTOKEN language NCName ID IDREF ENTITY hexBinary 	<p>Todos los niveles de correlación:</p> <p>char[</p> <p>z</p> <p>]</p>

Esquema simpleType	Tipos de datos C y C++
<pre><xsd:simpleType> <xsd:restriction base="xsd: type"> <xsd:length value=" z"/> </xsd:restriction> </xsd:simpleType></pre> <p>donde <i>type</i> es uno de:</p> <ul style="list-style-type: none"> string normalizedString token Name NMTOKEN language NCName ID IDREF ENTITY 	<p>Cuando CCSID=1200 está en el nivel de correlación 4.0 y superior:</p> <pre>char16_t[z]</pre>
<pre><xsd:simpleType> <xsd:restriction base="xsd: type"> </xsd:restriction> </xsd:simpleType></pre> <p>donde <i>type</i> es uno de:</p> <ul style="list-style-type: none"> duration date decimal time gDay gMonth gYear gMonthDay gYearMonth 	<p>Todos los niveles de correlación:</p> <pre>char[32]</pre>
<pre><xsd:simpleType> <xsd:restriction base="xsd:dateTime"> </xsd:restriction> </xsd:simpleType></pre>	<p>Nivel de correlación 1.2 e inferior:</p> <pre>char[32]</pre> <p>Nivel de correlación 2.0 y superior:</p> <pre>char[40]</pre> <p>Nivel de correlación 3.0 y superior:</p> <pre>char[8]</pre> <p>El formato de la indicación de fecha y hora es CICS ABSTIME.</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:byte"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos los niveles de correlación:</p> <pre>signed char</pre>

Esquema simpleType	Tipos de datos C y C++
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedByte"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos los niveles de correlación:</p> <p>char</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:short"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos los niveles de correlación:</p> <p>short</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedShort"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos los niveles de correlación:</p> <p>unsigned short</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:integer"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos los niveles de correlación:</p> <p>char[33]</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:int"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos los niveles de correlación:</p> <p>int</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedInt"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos los niveles de correlación:</p> <p>unsigned int</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:long"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos los niveles de correlación:</p> <p>long long</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedLong"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos los niveles de correlación:</p> <p>unsigned long long</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:boolean"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos los niveles de correlación:</p> <p>bool (sólo C++) short (sólo C)</p>

Esquema simpleType	Tipos de datos C y C++
<pre><xsd:simpleType> <xsd:list> <xsd:simpleType> <xsd:restriction base="xsd:int"/> </xsd:simpleType> </xsd:list> </xsd:simpleType></pre>	<p>Nivel de correlación 1.0:</p> <p>No soportado</p> <p>Nivel de correlación 1.1 y superior:</p> <p>char[255]</p>
<pre><xsd:simpleType> <xsd:union memberTypes="xsd:int xsd:string"/> </xsd:simpleType></pre>	<p>Nivel de correlación 1.0:</p> <p>No soportado</p> <p>Nivel de correlación 1.1 y superior:</p> <p>char[255]</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:base64Binary"> <xsd:length value=" z"/> </xsd:restriction> </xsd:simpleType> <xsd:simpleType> <xsd:restriction base="xsd:base64Binary"> </xsd:restriction> </xsd:simpleType></pre> <p>donde la longitud no está definida</p>	<p>Nivel de correlación 1.1 e inferior:</p> <p>char[y]</p> <p>donde $y = 4 \times (\text{ceil}(z/3))$. $\text{ceil}(x)$ es el entero más pequeño mayor que o igual a x.</p> <p>Nivel de correlación 1.2 y superior:</p> <p>char[z]</p> <p>donde la longitud es fija.</p> <p>char[16]</p> <p>es el nombre del contenedor que almacena los datos binarios cuando no se define la longitud.</p>

Esquema simpleType	Tipos de datos C y C++
<pre><xsd:simpleType> <xsd:restriction base="xsd:float"> </xsd:restriction> </xsd:simpleType></pre>	<p>Nivel de correlación 1.1 e inferior:</p> <p style="text-align: center;">char[32]</p> <p>Nivel de correlación 1.2 y superior:</p> <p style="text-align: center;">float(*)</p> <p>Nota: La representación de datos de coma flotante hexadecimal (HFP) de IBM no es exactamente lo mismo que la representación IEEE-754-1985 que se utiliza para XML. Algunos valores podrían no convertirse de forma exacta de una representación a la otra. Algunos valores sumamente grandes o pequeños podrían no ser válidos para los tipos de datos xsd:float. Algunos valores podrían perder precisión cuando se convierten a o desde la representación HFP. Si las conversiones precisas son importantes, considere sustituir el uso de tipos de datos flotantes por alternativas de precisión fija.</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:double"> </xsd:restriction> </xsd:simpleType></pre>	<p>Nivel de correlación 1.0 e inferior:</p> <p style="text-align: center;">char[32]</p> <p>Nivel de correlación 1.2 y superior:</p> <p style="text-align: center;">double(*)</p> <p>Nota: La representación de datos de coma flotante hexadecimal (HFP) de IBM no es exactamente lo mismo que la representación IEEE-754-1985 que se utiliza para XML. Algunos valores podrían no convertirse de forma exacta de una representación a la otra. Algunos valores sumamente grandes o pequeños podrían no ser válidos para los tipos de datos xsd:double. Algunos valores podrían perder precisión cuando se convierten a o desde la representación HFP. Si las conversiones precisas son importantes, considere sustituir el uso de tipos de datos dobles por alternativas de precisión fija.</p>

Correlación de PL/I con un esquema XML:

Los programas de utilidad DFHLS2SC y DFHLS2WS soportan correlaciones entre estructuras de datos de PL/I y definiciones de esquema XML. Puesto que el compilador de Enterprise PL/I y los compiladores de PL/I más antiguos difieren, se admiten dos opciones de lenguaje: PLI-ENTERPRISE y PLI-OTHER.

Los nombres PL/I se convierten en nombres XML según las reglas siguientes:

1. Los caracteres que no son válidos en los nombres de elemento XML se sustituyen por ' x '.
2. Los nombres duplicados se convierten en exclusivos añadiendo uno o varios dígitos numéricos.

Por ejemplo, dos instancias de year pasan a ser year y year1.

DFHLS2SC y DFHLS2WS correlacionan tipos de datos PL/I con elementos de esquema según la tabla siguiente. Los tipos PL/I que no se muestran en la tabla no son compatibles con DFHLS2SC o DFHLS2WS. También se aplican las siguientes restricciones:

- Los elementos de datos con el atributo COMPLEX no se soportan.
- Los elementos de datos con atributo FLOAT se soportan en un nivel de correlación de 1.2 o superior. Enterprise PL/I FLOAT IEEE no se soporta.
- Las series DBCS puras VARYING y VARYINGZ se soportan en un nivel de correlación de 1.2 o superior.
- Los elementos de datos que se especifican con DECIMAL(*p* , *q*) se soportan únicamente cuando $p \geq q$
- Los elementos de datos que se especifican con BINARY(*p* , *q*) se soportan únicamente cuando $q = 0$.
- Si se especifica el atributo PRECISION para un elemento de datos, se ignora.
- No se soportan las series PICTURE.
- Los elementos de datos ORDINAL se tratan como tipos de datos FIXED BINARY(7) .
- Si el parámetro **MAPPING-LEVEL** se establece en 1.2 o superior y el parámetro **CHAR-VARYING** se establece en NULL, las matrices de caracteres se correlacionan con una xsd:string y se procesan como series terminadas en nulo; esta correlación no se aplica a Enterprise PL/I.
- Si el parámetro **MAPPING-LEVEL** se establece en 1.2 o superior y el parámetro **CHAR-VARYING** se establece en BINARY, las correlaciones de caracteres se correlacionan con xsd:base64Binary y se procesan como datos binarios.
- Si el parámetro **MAPPING-LEVEL** se establece en 1.2 o superior y el parámetro **CHAR-VARYING** se establece en COLLAPSE, <xsd:whiteSpace value="collapse"/> se genera para las series.

DFHLS2SC y DFHLS2WS no implementan completamente los algoritmos de relleno de PL/I; por lo tanto, debe declarar bytes de relleno explícitamente en la estructura de datos. DFHLS2SC y DFHLS2WS emiten un mensajes si detectan que faltan los bytes de relleno. Cada estructura de nivel superior debe comenzar con un límite de palabra doble y cada byte de la estructura debe correlacionarse con el límite correcto. Tenga presente este fragmento de código:

```
3 FIELD1 FIXED BINARY(7),
3 FIELD2 FIXED BINARY(31),
3 FIELD3 FIXED BINARY(63);
```

En este ejemplo:

- FIELD1 tiene 1 byte de longitud y se puede alinear con cualquier límite.
- FIELD2 tiene 4 bytes de longitud y debe alinearse con un límite de palabra completa.
- FIELD3 tiene 8 bytes de longitud y debe alinearse con un límite de palabra doble.

El compilador Enterprise PL/I alinea los campos en el orden siguiente:

1. FIELD3 se alinea primero porque tiene los requisitos de límite más fuertes.
2. FIELD2 se alinea en un límite de palabra completa inmediatamente antes de FIELD3.
3. FIELD1 se alinea en un límite de byte inmediatamente antes de FIELD3.

Por último, para que toda la estructura se alinee en un límite de palabra completa, el compilador inserta tres bytes de relleno inmediatamente antes de FIELD1.

Puesto que DFHLS2WS no inserta bytes de relleno equivalentes, debe declararlos explícitamente antes de que DFHLS2WS procese la estructura. Por ejemplo:

```
3 PAD1 FIXED BINARY(7),
3 PAD2 FIXED BINARY(7),
3 PAD3 FIXED BINARY(7),
3 FIELD1 FIXED BINARY(7),
3 FIELD2 FIXED BINARY(31),
3 FIELD3 FIXED BINARY(63);
```

De forma alternativa, puede cambiar la estructura para declarar todos los campos como no alineados y volver a compilar la aplicación que utiliza la estructura. Para obtener más información sobre los requisitos de alineación de memoria estructural de PL/I, consulte *Enterprise PL/I Language Reference*.

Descripción de datos PL/I	Esquema
FIXED BINARY (<i>n</i>) donde $n \leq 7$	<code><xsd:simpleType> <xsd:restriction base="xsd:byte"/> </xsd:simpleType></code>
FIXED BINARY (<i>n</i>) donde $8 \leq n \leq 15$	<code><xsd:simpleType> <xsd:restriction base="xsd:short"/> </xsd:simpleType></code>
FIXED BINARY (<i>n</i>) donde $16 \leq n \leq 31$	<code><xsd:simpleType> <xsd:restriction base="xsd:int"/> </xsd:simpleType></code>
FIXED BINARY (<i>n</i>) donde $32 \leq n \leq 63$ Restricción: Sólo Enterprise PL/I	<code><xsd:simpleType> <xsd:restriction base="xsd:long"/> </xsd:simpleType></code>
UNSIGNED FIXED BINARY(<i>n</i>) donde $n \leq 8$ Restricción: Sólo Enterprise PL/I	<code><xsd:simpleType> <xsd:restriction base="xsd:unsignedByte"/> </xsd:simpleType></code>
UNSIGNED FIXED BINARY(<i>n</i>) donde $9 \leq n \leq 16$ Restricción: Sólo Enterprise PL/I	<code><xsd:simpleType> <xsd:restriction base="xsd:unsignedShort"/> </xsd:simpleType></code>
UNSIGNED FIXED BINARY(<i>n</i>) donde $17 \leq n \leq 32$ Restricción: Sólo Enterprise PL/I	<code><xsd:simpleType> <xsd:restriction base="xsd:unsignedInt"/> </xsd:simpleType></code>
UNSIGNED FIXED BINARY(<i>n</i>) donde $33 \leq n \leq 64$ Restricción: Sólo Enterprise PL/I	<code><xsd:simpleType> <xsd:restriction base="xsd:unsignedLong"/> </xsd:simpleType></code>
FIXED DECIMAL(<i>n</i> , <i>m</i>)	<code><xsd:simpleType> <xsd:restriction base="xsd:decimal"> <xsd:totalDigits value="n"/> <xsd:fractionDigits value="m"/> </xsd:restriction> </xsd:simpleType></code>

Descripción de datos PL/I	Esquema
<p>FIXED DECIMAL(15)</p> <p>Se soporta en el nivel de correlación 3.0 y superior cuando DATETIME=PACKED15</p>	<pre><xsd:simpleType> <xsd:restriction base="xsd:dateTime" </xsd:restriction> </xsd:simpleType></pre> <p>El formato de la indicación de fecha y hora es CICS ABSTIME.</p>
<p>BIT(<i>n</i>)</p> <p>donde <i>n</i> es un múltiplo de 8. No se soportan otros valores.</p>	<pre><xsd:simpleType> <xsd:restriction base="xsd:hexBinary"> <xsd:length value=" <i>m</i>" /> </xsd:restriction> </xsd:simpleType></pre> <p>donde $m = n / 8$</p>
<p>CHARACTER(<i>n</i>)</p> <p>VARYING y VARYINGZ también se soportan en el nivel de correlación 1.2 y superior.</p> <p>Restricción: VARYINGZ sólo se soporta en Enterprise PL/I</p>	<pre><xsd:simpleType> <xsd:restriction base="xsd:string"> <xsd:maxLength value=" <i>n</i>" /> <xsd:whiteSpace value="preserve" /> </xsd:restriction> </xsd:simpleType></pre>
<p>GRAPHIC(<i>n</i>)</p> <p>VARYING y VARYINGZ también se soportan en el nivel de correlación 1.2 y superior.</p> <p>Restricción: VARYINGZ sólo es compatible con Enterprise PL/I</p>	<p>En un nivel de correlación de 1.0 y 1.1, donde $m = 2 * n$:</p> <pre><xsd:simpleType> <xsd:restriction base="xsd:hexBinary"> <xsd:length value=" <i>m</i>" /> </xsd:restriction> </xsd:simpleType></pre> <p>En un nivel de correlación 1.2 o superior:</p> <pre><xsd:simpleType> <xsd:restriction base="xsd:string"> <xsd:length value=" <i>n</i>" /> <xsd:whiteSpace value="preserve" /> </xsd:restriction> </xsd:simpleType></pre>

Descripción de datos PL/I	Esquema
<p>WIDECHAR(n)</p> <p>Restricción: Sólo Enterprise PL/I</p>	<p>En un nivel de correlación de 1.0 y 1.1, donde $m = 2 * n$:</p> <pre><xsd:simpleType> <xsd:restriction base="xsd:hexBinary"> <xsd:length value=" m" /> </xsd:restriction> </xsd:simpleType></pre> <p>En un nivel de correlación 1.2 o superior:</p> <pre><xsd:simpleType> <xsd:restriction base="xsd:hexBinary"> <xsd:length value=" n" /> </xsd:restriction> </xsd:simpleType></pre> <p>En un nivel de correlación 4.0 y superior, CICS llena el campos de estructura de datos de aplicación con datos UTF-16.</p> <pre><xsd:simpleType> <xsd:restriction base="xsd:string"> <xsd:maxLength value="n" /> <xsd:whiteSpace value="preserve" /> </xsd:restriction> </xsd:simpleType></pre>
<p>ORDINAL</p> <p>Restricción: Sólo Enterprise PL/I</p>	<pre><xsd:simpleType> <xsd:restriction base="xsd:byte" /> </xsd:simpleType></pre>
<p>BINARY FLOAT(n)</p> <p>donde $n \leq 21$</p> <p>Se soporta en un nivel de correlación 1.2 y superiores.</p>	<pre><xsd:simpleType> <xsd:restriction base="xsd:float"> </xsd:restriction> </xsd:simpleType></pre>
<p>BINARY FLOAT(n)</p> <p>donde $21 < n \leq 53$</p> <p>No se soportan los valores superiores a 53.</p> <p>Se soporta en un nivel de correlación 1.2 y superiores.</p>	<pre><xsd:simpleType> <xsd:restriction base="xsd:double"> </xsd:restriction> </xsd:simpleType></pre>
<p>DECIMAL FLOAT(n)</p> <p>donde $n \leq 6$</p> <p>Se soporta en un nivel de correlación 1.2 y superiores.</p> <p>Nota: La representación de datos de coma flotante hexadecimal (HFP) de IBM no es exactamente lo mismo que la representación IEEE-754-1985 que se utiliza para XML. Algunos valores podrían no convertirse de forma exacta de una representación a la otra. Algunos valores sumamente grandes o pequeños podrían no ser válidos para los tipos de datos xsd:float. Algunos valores podrían perder precisión cuando se convierten a o desde la representación HFP. Si las conversiones precisas son importantes, considere sustituir el uso de tipos de datos DECIMAL FLOAT por alternativas de precisión fija.</p>	<pre><xsd:simpleType> <xsd:restriction base="xsd:float"> </xsd:restriction> </xsd:simpleType></pre>

Descripción de datos PL/I	Esquema
<p>DECIMAL FLOAT(<i>n</i>) donde $6 < n \leq 16$</p> <p>No se soportan los valores superiores a 16.</p> <p>Se soporta en un nivel de correlación 1.2 y superiores.</p> <p>Nota: La representación de datos de coma flotante hexadecimal (HFP) de IBM no es exactamente lo mismo que la representación IEEE-754-1985 que se utiliza para XML. Algunos valores podrían no convertirse de forma exacta de una representación a la otra. Algunos valores sumamente grandes o pequeños podrían no ser válidos para los tipos de datos <code>xsd:double</code>. Algunos valores podrían perder precisión cuando se convierten a o desde la representación HFP. Si las conversiones precisas son importantes, considere sustituir el uso de tipos de datos DECIMAL FLOAT por alternativas de precisión fija.</p>	<pre><xsd:simpleType> <xsd:restriction base="xsd:double"> </xsd:restriction> </xsd:simpleType></pre>

Esquema XML para correlación PL/I:

Los programas de utilidad DFHSC2LS y DFHWS2LS soportan la correlación entre definiciones de esquema XML y estructuras de datos PL/I. Puesto que el compilador de Enterprise PL/I y los compiladores de PL/I más antiguos difieren, se admiten dos opciones de lenguaje: PLI-ENTERPRISE y PLI-OTHER.

Reglas para la correlación de nombres de elemento del esquema para PL/I

Los asistentes de CICS generan nombres exclusivos y válidos para variables PL/I de los nombres de elemento del esquema utilizando las reglas siguientes:

1. Caracteres que no sean A-Z, a-z, 0-9, @, #, _ o \$ se sustituyen por ' X '.

Por ejemplo, `monthly-total` pasa a ser `monthlyXtotal`.

Puede utilizar el parámetro **MAPPING-OVERRIDES** para cambiar la forma en la que se manejan otros caracteres. Por ejemplo, si establece el valor **HYPHENS-AS-UNDERSCORES**, cualquier guión de XML se convierte en un guión bajo en lugar de en una X. Por ejemplo, `monthly-total` pasa a ser `monthly_total`.

2. Si el esquema especifica que la variable tiene cardinalidad variable (es decir, se especifican los atributos `minOccurs` y `maxOccurs` con valores diferentes en `xsd:element`), y el nombre de elemento de esquema es mayor de 24 caracteres, se trunca en esa longitud.

Si el esquema especifica que la variable tiene cardinalidad fija y el nombre de elemento de esquema es mayor de 29 caracteres, se trunca en esa longitud.

3. Los nombres duplicados en el mismo ámbito se convierten en exclusivos añadiendo uno o varios dígitos numéricos a la segunda y siguientes instancias del nombre.

Por ejemplo, tres instancias de `year` pasan a ser `year`, `year1` y `year2`.

4. Se reservan cinco caracteres para las series `_cont` o `_num`, que se utilizan cuando el esquema especifica que la variable tiene una cardinalidad variable; es decir, cuando los atributos `minOccurs` y `maxOccurs` se especifican con valores diferentes.

Para obtener más información, consulte el apartado “Matrices de variables de elementos” en la página 349.

5. Para los atributos, se aplican las reglas anteriores al elemento. Se añade el prefijo `attr-` al nombre de elemento y va seguido de `-value` o `-exist`. Si la longitud total es mayor de 28 caracteres, el nombre de elemento se trunca. Para obtener más información, consulte el apartado “Soporte para atributos XML” en la página 354.

El atributo `nillable` tiene reglas especiales. Se añade el prefijo `attr-`, pero también se añade `nil-` al principio del nombre de elemento. El nombre de elemento va seguido de `-value`. Si la longitud total es mayor de 28 caracteres, el nombre de elemento se trunca.

La longitud total del nombre resultante es de 31 caracteres o menos.

Reglas para la correlación de tipos de esquema para PL/I

DFHSC2LS y DFHWS2LS correlacionan tipos de esquema con tipos de datos PL/I según la tabla siguiente. También tenga en cuenta los siguientes puntos:

- Si el parámetro **MAPPING-LEVEL** se establece en 1.2 o superior y el parámetro **CHAR-VARYING** se establece en `NULL`, los datos de caracteres de longitud variable se correlacionan con series terminadas en nulo y se asigna un carácter adicional al terminador en nulo.
- Si el parámetro **MAPPING-LEVEL** se establece en 1.2 o superior y el parámetro **CHAR-VARYING** no se especifica, de forma predeterminada, los datos de caracteres de longitud variable se correlacionan con un tipo de datos `VARYINGZ` para Enterprise PL/I y el tipo de datos `VARYING` para Otros PL/I.
- Los datos binarios de longitud variables se correlacionan con un tipo de datos `VARYING` si es menor de 32 768 bytes y con un contenedor si es mayor de 32 768 bytes.

Esquema	Descripción de datos PL/I
<pre><xsd:simpleType> <xsd:restriction base="xsd:anyType"> </xsd:restriction> </xsd:simpleType></pre>	<p>Nivel de correlación 2.0 e inferior:</p> <p>No soportado</p> <p>Nivel de correlación 2.1 y superior:</p> <p>Soportado</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:anySimpletype"> </xsd:restriction> </xsd:simpleType></pre>	<p>Nivel de correlación 1.1 y superior: <code>CHAR(255)</code></p>

Esquema	Descripción de datos PL/I
<pre><xsd:simpleType> <xsd:restriction base="xsd: type"> <xsd:maxLength value=" z"/> <xsd:whiteSpace value="preserve"/> </xsd:restriction> </xsd:simpleType></pre> <p>donde <i>type</i> es uno de:</p> <ul style="list-style-type: none"> string normalizedString token Name NMTOKEN language NCName ID IDREF ENTITY 	<p>Todos los niveles de correlación: CHARACTER(z)</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd: type"> <xsd:length value=" z"/> </xsd:restriction> </xsd:simpleType></pre> <p>donde <i>type</i> es uno de:</p> <ul style="list-style-type: none"> string normalizedString token Name NMTOKEN language NCName ID IDREF ENTITY 	<p>Cuando CCSID=1200 está en el nivel de correlación 4.0 y superior:</p> <p style="text-align: center;">WIDECHAR(</p> <p style="text-align: center;">z</p> <p style="text-align: center;">)</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd: type"> </xsd:restriction> </xsd:simpleType></pre> <p>donde <i>type</i> es uno de:</p> <ul style="list-style-type: none"> duration date time gDay gMonth gYear gMonthDay gYearMonth 	<p>Todos los niveles de correlación: CHAR(32)</p>

Esquema	Descripción de datos PL/I
<pre><xsd:simpleType> <xsd:restriction base="xsd:dateTime"> </xsd:restriction> </xsd:simpleType></pre>	<p>Nivel de correlación 1.2 e inferior:</p> <p>CHAR(32)</p> <p>Nivel de correlación 2.0 y superior:</p> <p>CHAR(40)</p> <p>Nivel de correlación 3.0 y superior:</p> <p>FIXED DECIMAL(15)</p> <p>El formato de la indicación de fecha y hora es CICS ABSTIME.</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:hexBinary"> <xsd:length value=" y"/> </xsd:restriction> </xsd:simpleType></pre>	<p>Nivel de correlación 1.1 e inferior:</p> <p>BIT(z)</p> <p>donde $z = 8 \times y$ y $z < 4095$ bytes.</p> <p>CHAR(z)</p> <p>donde $z = 8 \times y$ y $z > 4095$ bytes.</p> <p>Niveles de correlación 1.2 y superior:</p> <p>CHAR(y)</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:byte"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos los niveles de correlación:</p> <p>Enterprise PL/I SIGNED FIXED BINARY (7)</p> <p>Otros PL/I FIXED BINARY (7)</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedByte"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos los niveles de correlación:</p> <p>Enterprise PL/I UNSIGNED FIXED BINARY (8)</p> <p>Otros PL/I FIXED BINARY (8)</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:short"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos los niveles de correlación:</p> <p>Enterprise PL/I SIGNED FIXED BINARY (15)</p> <p>Otros PL/I FIXED BINARY (15)</p>

Esquema	Descripción de datos PL/I
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedShort"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos los niveles de correlación:</p> <p>Enterprise PL/I UNSIGNED FIXED BINARY (16)</p> <p>Otros PL/I FIXED BINARY (16)</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:integer"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos los niveles de correlación:</p> <p>Enterprise PL/I FIXED DECIMAL(31,0)</p> <p>Otros PL/I FIXED DECIMAL(15,0)</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:int"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos los niveles de correlación:</p> <p>Enterprise PL/I SIGNED FIXED BINARY (31)</p> <p>Otros PL/I FIXED BINARY (31)</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedInt"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos los niveles de correlación:</p> <p>Enterprise PL/I UNSIGNED FIXED BINARY(32)</p> <p>Otros PL/I BIT(64)</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:long"> </xsd:restriction> </xsd:simpleType></pre>	<p>Nivel de correlación 1.1 e inferior:</p> <p>Enterprise PL/I SIGNED FIXED BINARY(63)</p> <p>Nivel de correlación 1.2 y superior:</p> <p>Enterprise PL/I CHAR(y) donde y es una longitud fija menor de 16 MB.</p> <p>Todos los niveles de correlación:</p> <p>Otros PL/I BIT(64)</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedLong"> </xsd:restriction> </xsd:simpleType></pre>	<p>Nivel de correlación 1.1 e inferior:</p> <p>Enterprise PL/I UNSIGNED FIXED BINARY(64)</p> <p>Nivel de correlación 1.2 y superior:</p> <p>Enterprise PL/I CHAR(y) donde y es una longitud fija menor de 16 MB.</p> <p>Todos los niveles de correlación:</p> <p>Otros PL/I BIT(64)</p>

Esquema	Descripción de datos PL/I
<pre><xsd:simpleType> <xsd:restriction base="xsd:boolean"> </xsd:restriction> </xsd:simpleType></pre>	<p>Nivel de correlación 1.1 e inferior:</p> <p>Enterprise PL/I SIGNED FIXED BINARY (7)</p> <p>Otros PL/I FIXED BINARY (7)</p> <p>Nivel de correlación 1.2 y superior:</p> <p>Enterprise PL/I BIT(7) BIT(1)</p> <p>Otros PL/I BIT(7) BIT(1)</p> <p>donde se proporciona BIT(7) para la alineación y BIT(1) contiene el valor correlacionado booleano.</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:decimal"> <xsd:totalDigits value=" n"/> <xsd:fractionDigits value=" m"/> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos los niveles de correlación: FIXED DECIMAL(<i>n</i> , <i>m</i>)</p>
<pre><xsd:simpleType> <xsd:list> <xsd:simpleType> <xsd:restriction base="xsd:int"/> </xsd:simpleType> </xsd:list> </xsd:simpleType></pre>	<p>Todos los niveles de correlación: CHAR(255)</p>
<pre><xsd:simpleType> <xsd:union memberTypes="xsd:int xsd:string"/> </xsd:simpleType></pre>	<p>Todos los niveles de correlación: CHAR(255)</p>

Esquema	Descripción de datos PL/I
<pre> <xsd:simpleType> <xsd:restriction base="xsd:base64Binary"> <xsd:length value=" y"/> </xsd:restriction> </xsd:simpleType> <xsd:simpleType> <xsd:restriction base="xsd:base64Binary"> </xsd:restriction> </xsd:simpleType> </pre> <p>donde la longitud no está definida</p>	<p>Nivel de correlación 1.0:</p> <p>No soportado</p> <p>Nivel de correlación 1.1:</p> <p style="text-align: right;">CHAR(z)</p> <p>donde $z = 4 \times (\text{ceil}(y/3))$. $\text{ceil}(x)$ es el entero más pequeño mayor que o igual a x.</p> <p>Nivel de correlación 1.2 y superior:</p> <p style="text-align: right;">CHAR(y)</p> <p>donde la longitud es fija.</p> <p style="text-align: right;">CHAR(16)</p> <p>donde la longitud no está definida. El campo contiene el nombre de 16 bytes del contenedor que almacena los datos binarios.</p>
<pre> <xsd:simpleType> <xsd:restriction base="xsd:float"> </xsd:restriction> </xsd:simpleType> </pre>	<p>Niveles de correlación 1.0 y 1.1:</p> <p style="text-align: right;">CHAR(32)</p> <p>Nivel de correlación 1.2 y superior:</p> <p>Enterprise PL/I DECIMAL FLOAT(6) HEXADEC</p> <p>Otros PL/I DECIMAL FLOAT(6)</p> <p>Nota: La representación de datos de coma flotante hexadecimal (HFP) de IBM no es exactamente lo mismo que la representación IEEE-754-1985 que se utiliza para XML. Algunos valores podrían no convertirse de forma exacta de una representación a la otra. Algunos valores sumamente grandes o pequeños podrían no ser válidos para los tipos de datos <code>xsd:float</code>. Algunos valores podrían perder precisión cuando se convierten a o desde la representación HFP. Si las conversiones precisas son importantes, considere sustituir el uso de tipos de datos DECIMAL FLOAT por alternativas de precisión fija.</p>

Esquema	Descripción de datos PL/I
<pre><xsd:simpleType> <xsd:restriction base="xsd:double"> </xsd:restriction> </xsd:simpleType></pre>	<p>Niveles de correlación 1.0 y 1.1:</p> <p>CHAR(32)</p> <p>Nivel de correlación 1.2 y superior:</p> <p>Enterprise PL/I DECIMAL FLOAT(16) HEXADEC</p> <p>Otros PL/I DECIMAL FLOAT(16)</p> <p>Nota: La representación de datos de coma flotante hexadecimal (HFP) de IBM no es exactamente lo mismo que la representación IEEE-754-1985 que se utiliza para XML. Algunos valores podrían no convertirse de forma exacta de una representación a la otra. Algunos valores sumamente grandes o pequeños podrían no ser válidos para los tipos de datos xsd:double. Algunos valores podrían perder precisión cuando se convierten a o desde la representación HFP. Si las conversiones precisas son importantes, considere sustituir el uso de tipos de datos DECIMAL FLOAT por alternativas de precisión fija.</p>

Matrices de variables de elementos:

XML puede contener una matriz con distintos números de elementos. En general, los documentos WSDL y esquemas XML que contienen varios números de elementos no se correlacionan de forma eficiente en una única estructura de datos de lenguaje de alto nivel. CICS utiliza correlaciones basadas en contenedores o correlaciones en línea para gestionar varios elementos en XML.

Una matriz con varios elementos se representa en el esquema XML utilizando los atributos minOccurs y maxOccurs en la declaración del elemento:

- El atributo minOccurs especifica el número de veces mínimo que se puede dar el elemento. Puede tener un valor de 0 o un entero positivo.
- El atributo maxOccurs especifica el número de veces máximo que se puede dar el elemento. Puede tener un valor de un entero positivo mayor o igual al valor del atributo minOccurs. También puede tomar un valor de unbounded, que indica que no se aplica límite superior al número de veces que se puede dar el elemento.
- El valor predeterminado para ambos atributos es 1.

Este ejemplo indica una serie de 8 bytes que es opcional; es decir, que puede que no ocurra nunca o que ocurra una vez en el mensaje SOAP o XML de aplicación:

```
<xsd:element name="component"
minOccurs="0" maxOccurs="1">
<xsd:simpleType>
<xsd:restriction base="xsd:string">
<xsd:length value="8"/>
</xsd:restriction>
</xsd:simpleType>
</xsd:element>
```

El ejemplo siguiente indica una serie de 8 bytes que debe tener lugar al menos una vez:

```

<xsd:element name="component"
minOccurs="1" maxOccurs="unbounded">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:length value="8"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>

```

En general, los documentos WSDL que contienen varios números de elementos no se correlacionan de forma eficiente en una única estructura de datos de lenguaje de alto nivel. Por lo tanto, para gestionar estos casos, CICS utiliza una serie de estructuras de datos conectadas que se pasan al programa de aplicación en una serie de contenedores. Estas estructuras se utilizan como entrada y salida de la aplicación:

- Cuando CICS transforma XML en datos de aplicación, llena estas estructuras con los datos de aplicación y la aplicación los lee.
- Cuando CICS transforma los datos de aplicación en XML, lee los datos de aplicación en las estructuras que ha estado llenando la aplicación.

El formato de estas estructuras de datos se explica mejor con una serie de ejemplos. El XML puede ser de un mensaje SOAP o de una aplicación. Estos ejemplos utilizan una matriz de campos simples de 8 bytes. Sin embargo, el modelo soporta matrices de tipos de datos complejos y matrices de tipos de datos que contienen otras matrices.

Número fijo de elementos

El primer ejemplo ilustra un elemento que se produce tres veces exactamente:

```

<xsd:element name="component"
minOccurs="3" maxOccurs="3">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:length value="8"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>

```

En este ejemplo, debido a que el número de veces que se da un elemento se conoce de antemano, se puede representar como una matriz de longitud fija en una sencilla declaración COBOL (o el equivalente en otros lenguajes):

```
05 component PIC X(8) OCCURS 3 TIMES
```

Número variado de elementos en el nivel de correlación 2 e inferior

Este ejemplo ilustra un elemento obligatorio que puede encontrarse de una a cinco veces:

```

<xsd:element name="component"
minOccurs="1" maxOccurs="5">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:length value="8"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>

```

La estructura de datos principal contiene una declaración de dos campos. Cuando CICS transforma el XML en datos binarios, el primer campo component-num

contiene el número de veces que aparece el elemento en el XML, y el segundo campo, `component-cont`, contiene el nombre del contenedor:

```
05 component-num PIC S9(9) COMP-5  
05 component-cont PIC X(16)
```

Una segunda estructura de datos contiene la declaración del elemento:

```
01 DFHWS-component  
02 component PIC X(8)
```

Debe examinar el valor de `component-num` (que contendrá un valor en el rango de 1 a 5) para averiguar cuántas veces se da el elemento. El contenido del elemento está en el contenedor denominado `component-cont`; el contenedor tiene una matriz de elementos, donde cada elemento está correlacionado por la estructura de datos `DFHWS-component`.

Si `minOccurs="0"` y `maxOccurs="1"`, el elemento es opcional. Para procesar la estructura de datos en el programa de aplicación, debe examinar el valor de `component-num`:

- Si es cero, el mensaje no tiene elemento de componente y el contenido de `component-cont` no está definido.
- Si es uno, el elemento de componente está en el contenedor denominado `component-cont`.

El contenido del contenedor está correlacionado por la estructura de datos `DFHWS-component`.

Nota: Si el mensaje SOAP consta de un único elemento recurrente, `DFHWS2LS` genera dos estructuras de lenguaje. La estructura de lenguaje principal contiene el número de elementos en la matriz y el nombre del contenedor que tiene la matriz de elementos. La segunda estructura de lenguaje correlaciona una única instancia de elemento recurrente.

Número variado de elementos en el nivel de correlación 2.1 y superior

En el nivel de correlación 2.1 y superior, puede utilizar el parámetro **INLINE-MAXOCCURS-LIMIT** en los asistentes de CICS. El parámetro **INLINE-MAXOCCURS-LIMIT** especifica la forma en la que se manejan varios elementos. Las opciones de correlación para un número variado de elementos son correlaciones basadas en contenedores, descritas en “Número variado de elementos en el nivel de correlación 2 e inferior” en la página 350, o correlaciones en línea. El *valor* de este parámetro puede ser un entero positivo en el rango de 0 a 32767:

- El valor predeterminado de **INLINE-MAXOCCURS-LIMIT** es 1, que garantiza que los elementos opcionales están correlacionados en línea.
- Un valor de 0 para el parámetro **INLINE-MAXOCCURS-LIMIT** impide la correlación en línea.
- Si `maxOccurs` es menor o igual al valor de **INLINE-MAXOCCURS-LIMIT**, se utiliza la correlación en línea.
- Si `maxOccurs` es mayor que el valor de **INLINE-MAXOCCURS-LIMIT**, se utiliza la correlación basada en contenedores.

La correlación de un número variado de elementos en línea da como resultado la generación de una matriz, como sucede con el ejemplo de aparición fija anterior, y un contador. El campo `component-num` indica cuántas instancias del elemento están presentes y si están señaladas por una matriz. Para el ejemplos mostrado en

“Número variado de elementos en el nivel de correlación 2 e inferior” en la página 350, cuando **INLINE-MAXOCCURS-LIMIT** es menor o igual a 5, la estructura de datos generada es como ésta:

```
05 component-num PIC S9(9) COMP-5 SYNC.  
05 component OCCURS 5 PIC X(8).
```

El primer campo, component-num, es idéntico a la salida para el ejemplo de correlación basada en contenedores de la sección anterior. El segundo campo contiene una matriz de longitud 5 que es lo suficientemente larga para contener el número máximo de elementos que puede generar.

La correlación en línea difiere de la correlación basada en contenedores, que almacena el número de apariciones del elemento y el nombre del contenedor donde están colocados los datos, porque almacena todos los datos en el contenedor actual. Almacenar los datos en el contenedor actual normalmente mejorará el rendimiento y hará que la correlación en línea sea preferible.

Matrices de variables anidadas

Los esquemas XML y documentos WSDL complejos pueden contener elementos recurrentes variables, que a su vez contienen elementos recurrentes variables. En este caso, la estructura descrita se amplía más allá de los dos niveles descritos en los ejemplos.

Este ejemplo ilustra en elemento opcional denominado <component2> que se anida en un elemento obligatorio denominado <component1>, donde el elemento obligatorio puede encontrarse de una a cinco veces:

```
<xsd:element name="component1"  
minOccurs="1" maxOccurs="5">  
  <xsd:complexType>  
    <xsd:sequence>  
      <xsd:element name="component2"  
minOccurs="0" maxOccurs="1">  
        <xsd:simpleType>  
          <xsd:restriction base="xsd:string">  
            <xsd:length value="8"/>  
          </xsd:restriction>  
        </xsd:simpleType>  
      </xsd:element>  
    </xsd:sequence>  
  </xsd:complexType>  
</xsd:element>
```

La estructura de datos de nivel superior es exactamente la misma que en los ejemplos anteriores:

```
05 component1-num PIC S9(9) COMP-5  
05 component1-cont PIC X(16)
```

Sin embargo, la segunda estructura de datos contiene estos elementos:

```
01 DFHWS-component1  
02 component2-num PIC S9(9) COMP-5  
02 component2-cont PIC X(16)
```

Una estructura de tercer nivel contiene estos elementos:

```
01 DFHWS-component2  
02 component2 PIC X(8)
```


El número de apariciones del elemento más externo "component1" está en component1-num.

El contenedor nombrado en component1-cont contiene una matriz con varias instancias de la segunda estructura de datos DFHWS-component1.

Cada instancia de component2-cont nombra un contenedor diferente, cada uno de los cuales contiene la estructura de datos correlacionada por la estructura de tercer nivel DFHWS-component2.

Para ilustrar esta estructura, tenga en cuenta el fragmento de XML que coincide con el ejemplo:

```
<component1><component2>
string1
</component2></component1>
<component1><component2>
string2
</component2></component1>
<component1></component1>
```

<component1> se da tres veces. Las dos primeras contienen cada una una instancia de <component2>; la tercera instancia no.

En la estructura de datos de nivel superior, component1-num contiene un valor de 3. El contenedor nombrado en component1-cont tiene tres instancias de DFHWS-component1 :

1. En la primera, component2-num tiene un valor de 1, y el contenedor nombrado en component2-cont contiene *string1*.
2. En la segunda, component2-num tiene un valor de 1, y el contenedor nombrado en component2-cont contiene *string2*.
3. En la tercera, component2-num tiene un valor de 0, y el contenido de component2-cont no está definido.

En esta instancia, la estructura de datos completa se representa por medio de cuatro contenedores en total:

- La estructura de datos raíz en el contenedor DFHWS-DATA
- El contenedor nombrado en component1-cont
- Los dos contenedores nombrados en las dos primeras instancias de component2-cont.

Estructuras opcionales y xsd:choice

DFHWS2LS y DFHSC2LS soportan el uso de maxOccurs y minOccurs en elementos <xsd:sequence>, <xsd:choice>, y <xsd:all> sólo en el nivel de correlación 2.1 y superior, donde los atributos minOccurs y maxOccurs están configurados en minOccurs="0" y maxOccurs="1".

Los asistentes generan correlaciones que tratan estos elementos como si cada elemento hijo en ellos fuera opcional. Cuando implementa una aplicación con estos elementos, asegúrese de que la aplicación no genera combinaciones no válidas de opciones. Cada uno de los elementos tiene su propio campo count en la estructura de lenguajes generada, estos campos deben establecerse todos en "0" o todos en "1". Cualquier otra combinación de valores no es válida; excepto con elementos <xsd:choice>.

Los elementos <xsd:choice> indican que sólo se puede utilizar una de las opciones del elemento. Esto se soporta en todos los niveles de correlación. Los asistentes gestionan cada una de las opciones en un <xsd:choice> aunque es un elemento <xsd:sequence> con minOccurs="0" y maxOccurs="1". Tenga cuidado cuando implementa una aplicación utilizando el elemento <xsd:choice> para asegurarse de que la aplicación no genera combinaciones no válidas de las opciones. Cada uno de los elementos tiene su propio campo count en la estructura de lenguajes generada, exactamente uno debe estar establecido en '1' y los demás establecidos en '0'. Cualquier otra combinación de valores no es válida, excepto cuando el elemento <xsd:choice> es opcional, en cuyo caso es válido para todos los campos que esté establecido en '0'.

Soporte para atributos XML:

Los esquemas XML pueden especificar atributos permitidos o necesarios en XML. Los programas de utilidad del asistente de CICS, DFHWS2LS y DFHSC2LS, ignoran los atributos XML de forma predeterminada. Para procesar atributos XML que se definen en el esquema XML, el valor del parámetro **MAPPING-LEVEL** debe ser 1.1 o superior.

Atributos opcionales

Los atributos pueden ser opcionales o necesarios y se pueden asociar a cualquier elemento en el mensaje SOAP o XML para una aplicación. Para cada atributo opcional definido en el esquema, se generan dos campos en la estructura de lenguaje adecuada:

1. Un indicador de existencia; este campo se trata como un tipo de datos booleano y, normalmente, tiene 1 byte de longitud.
2. Un valor; este campo se correlaciona de la misma forma que un elemento XML de un tipo equivalentemente. Por ejemplo, un atributo de tipo NMTOKEN se correlaciona de la misma forma que un elemento XML de tipo NMTOKEN.

Los campos de valor y existencia del atributo aparecen en la estructura de lenguaje generada antes del campo para el elemento al que están asociados. Los atributos inesperado que aparecen en el documento de instancia se ignoran.

Por ejemplo, tenga en cuenta la siguiente definición de atributo de esquema:

```
<xsd:attribute name="age" type="xsd:short"
use="optional" />
```

Este atributo opcional se correlaciona con la siguiente estructura COBOL:

```
05 attr-age-exist PIC X DISPLAY
05 attr-age-value PIC S9999 COMP-5 SYNC
```

Proceso de tiempo de ejecución de atributos opcionales

El siguiente proceso de tiempo de ejecución tiene lugar para los atributos opcionales:

- Si el atributo está presente, se establece el indicador de existencia y el valor se correlaciona.
- Si el atributo no está presente, el indicador de existencia no se establece.
- Si el atributo tiene un valor predeterminado y está presente, el valor se correlaciona.
- Si el atributo tiene un valor predeterminado y no está presente, el valor predeterminado se correlaciona.

Los atributos opcionales que tienen valores predeterminados se tratan como atributos necesarios.

Cuando CICS transforma los datos en XML, tiene lugar el siguiente proceso de tiempo de ejecución:

- Si se establece el indicador de existencia, el atributo se transforma y se incluye en el XML.
- Si el distintivo de existencia no se establece, el atributo no se incluye en el XML.

Atributos necesarios y proceso de tiempo de ejecución

Para cada atributo necesario, sólo se genera el campo de valor en la estructura de lenguaje adecuada.

Si el atributo está presente en el XML, el valor se correlaciona. Si el atributo no está presente, se produce el siguiente proceso:

- Si la aplicación es un proveedor de servicios web, CICS genera un mensaje de error SOAP que indica que hay un error en el mensaje SOAP de cliente.
- Si la aplicación es un solicitante de servicio web, CICS emite un mensaje y devuelve una respuesta de error de conversión con un código RESP2 de 13 a la aplicación.
- Si la aplicación está utilizando el mandato **TRANSFORM XMLTODATA**, CICS emite un mensaje y devuelve una respuesta de solicitud no válida con un código RESP2 de 3 a la aplicación.

Cuando CICS produce un mensaje SOAP basado en el contenido de un COMMAREA o un contenedor, el atributo se transforma y se incluye en el mensaje. Cuando una aplicación utiliza el mandato **TRANSFORM DATATOXML**, CICS también transforma el atributo y lo incluye en el XML.

El atributo nillable

El atributo nillable es un atributo especial que puede aparecer en un `xsd:element` en un esquema XML. Especifica que el atributo `xsi:nil` es válido para el elemento en el XML. Si un elemento tiene especificado el atributo `xsi:nil`, indica que el elemento está presente pero no tiene valor y, por lo tanto, no tiene contenido asociado a él.

Si un esquema XML ha definido el atributo nillable como verdadero, se correlaciona como un atributo necesario que toma un valor booleano.

Cuando CICS recibe un mensaje SOAP o tiene que transformar XML para una aplicación que contiene un atributo `xsi:nil`, el valor del atributo es verdadero o falso. Si el valor es verdadero, la aplicación debe ignorar los valores del elemento o elementos anidados en el ámbito del atributo `xsi:nil`.

Cuando CICS produce un mensaje SOAP o XML basado en el contenido de un COMMAREA o de un contenedor para el que el valor para el atributo `xsi:nil` es verdadero, se produce el siguiente proceso:

- El atributo `xsi:nil` se genera en el XML o mensaje SOAP.
- El valor del elemento asociado se ignora.
- Cualquier elemento anidado dentro del elemento se ignora.

Ejemplo de mensaje SOAP

Tenga en cuenta el siguiente esquema XML de ejemplo, que puede ser parte de un documento WSDL:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="root" nillable="true">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element nillable="true" name="num" type="xsd:int" maxOccurs="3"
          minOccurs="3"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Aquí tiene un ejemplo de un mensaje SOAP parcial que se ajusta a este esquema:

```
<root xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <num xsi:nil="true"/>
  <num>15</num>
  <num xsi:nil="true"/>
</root>
```

En COBOL, este mensaje SOAP se correlaciona con estos elementos:

```
05 root
10 attr-nil-root-value PIC X DISPLAY
10 num OCCURS 3
15 num1 PIC S9(9) COMP-5 SYNC
15 attr-nil-num-value PIC X DISPLAY
10 filler PIC X(3)
```

Soporte para <xsd:any> y xsd:anyType:

DFHWS2LS y DFHSC2LS soportan el uso de <xsd:any> y xsd:anyType en el esquema XML. Puede utilizar el elemento de esquema XML <xsd:any> para describir una sección de un documento XML con contenido no definido. xsd:anyType es el tipo de datos base desde el que se derivan todos los tipos de datos simples y complejos; no tiene restricciones en el contenido de los datos.

Antes de poder utilizar <xsd:any> y xsd:anyType con los asistentes de CICS, establezca los siguientes parámetros:

- Establezca el parámetro **MAPPING-LEVEL** en 2.1 o superior.
- Para una aplicación de proveedor de servicios web, establezca el parámetro **PGMINT** en CHANNEL.

Ejemplo de <xsd:any>

Este ejemplo utiliza un elemento <xsd:any> para describir contenido XML no construido opcional después de la etiqueta "Surname" en la etiqueta "Customer":

```
<xsd:element name="Customer">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Title" type="xsd:string"/>
      <xsd:element name="FirstName" type="xsd:string"/>
      <xsd:element name="Surname" type="xsd:string"/>
      <xsd:any minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Un mensaje SOAP de ejemplo que se ajusta a este esquema XML es:

```

<xml version='1.0' encoding='UTF-8'?>
  <SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  >
    <SOAP-ENV:Body>
      <Customer xmlns="http://www.example.org/anyExample">
        <Title xmlns="">Mr</Title>
        <FirstName xmlns="">John</FirstName>
        <Surname xmlns="">Smith</Surname>
        <ExtraInformation xmlns="http://www.example.org/ExtraInformation">
          <!-- This 'ExtraInformation' tag is associated with the optional xsd:any
            from the XML schema.
            It can contain any well formed XML. -->
          <ExampleField1>one</ExampleField1>
          <ExampleField2>two</ExampleField2>
        </ExtraInformation>
      </Customer>
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>

```

Si este mensaje SOAP se envía a CICS , CICS llena el contenedor Customer-xml-cont con los siguientes datos XML:

```

<ExtraInformation
  xmlns="http://www.example.org/ExtraInformation">
  <!-- This 'ExtraInformation' tag is associated with the optional xsd:any
    from the XML schema.
    It can contain any well formed XML. -->
  <ExampleField1>one</ExampleField1>
  <ExampleField2>two</ExampleField2>
</ExtraInformation>

```

CICS también llena el contenedor Customer-xmlns-cont con las siguientes declaraciones de espacio de nombres XML que está en el ámbito; estas declaraciones están separadas por un espacio:

```

xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns="http://www.example.org/anyExample"

```

Ejemplo de xsd:anyType

El xsd:anyType es el tipo de datos base del que se derivan todos los tipos de datos simples y complejos. No restringe el contenido de datos. Si no especifica un tipo de datos, utiliza los valores predeterminados de xsd:anyType; por ejemplo, estos dos fragmentos XML son equivalentes:

```

<xsd:element name="Name" type="xsd:anyType"/>
<xsd:element name="Name"/>

```

Estructuras de lenguaje generadas

Las estructuras de lenguaje generadas para <xsd:any> o xsd:anyType toman el formato siguiente en COBOL y un formato equivalente para otros lenguajes:

elementName-xml-cont PIC X(16)

El nombre de un contenedor que tiene el XML sin procesar. Cuando CICS procesa un mensaje SOAP de entrada, coloca el subconjunto del mensaje SOAP que define <xsd:any> o xsd:anyType en este contenedor. La aplicación puede procesar los datos XML sólo de forma nativa. La aplicación debe generar el XML, llenar este contenedor y proporcionar el nombre de contenedor.

Este contenedor debe llenarse en modalidad de texto. Si CICS llena este contenedor, lo hace utilizando la misma variante de EBCDIC que el servicio web tiene definida para utilizar. Los caracteres que no existen en la

página de códigos EBCDIC de destino se sustituyen por caracteres sustitutos; aunque el contenedor lo lea una aplicación en UTF-8.

elementName-xmlns-cont PIC X(16)

El nombre de un contenedor que tiene cualquier declaración de prefijo de espacio de nombres que esté en el ámbito. El contenido de este contenedor es similar al contenido del contenedor DFHWS-XMLNS, excepto que incluye todas las declaraciones de espacio de nombres que están en el ámbito y que son relevantes, en lugar de sólo el subconjunto de la etiqueta Sobre SOAP.

Este contenedor debe llenarse en modalidad de texto. Si CICS llena este contenedor, lo hace utilizando la misma variante de EBCDIC que el servicio web tiene definida para utilizar. Los caracteres que no existen en la página de códigos EBCDIC de destino se sustituyen por caracteres sustitutos; aunque el contenedor lo lea una aplicación en UTF-8.

Este contenedor se utiliza sólo cuando los mensajes SOAP de proceso se envían a CICS. Si la aplicación intenta proporcionar un contenedor con declaraciones de espacio de nombres cuando se genera un mensaje SOAP de salida, CICS ignora el contenedor y su contenido. CICS requiere que el XML proporcionado por la aplicación sea completamente autocontenido con respecto a las declaraciones de espacio de nombres.

El nombre del elemento XML que contiene el elemento `<xsd:any>` se incluye en los nombres de variable que se generan para el elemento `<xsd:any>`. En el ejemplo `<xsd:any>`, el elemento `<xsd:any>` se anida dentro del elemento `<xsd:element name="Customer">` y los nombres de variables que se generan para el elemento `<xsd:any>` son `Customer-xml-cont PIC X(16)` y `Customer-xmlns-cont PIC X(16)`.

Para un tipo `xsd:anyType`, se utiliza el nombre de elemento XML directo; en el ejemplo `xsd:anyType` anterior, los nombres de variable son `Name-xml-cont PIC X(16)` y `Name-xmlns-cont PIC X(16)`.

Soporte para `<xsd:choice>`:

Un elemento `<xsd:choice>` indica que sólo se puede utilizar una de las opciones del elemento. Los asistentes de CICS proporcionan varios grados de soporte para elementos `<xsd:choice>` a distintos niveles de correlación.

Soporte para `<xsd:choice>` a nivel de correlación 2.2 y superior

En el nivel de correlación 2.2 y, DFHWS2LS y DFHSC2LS proporcionan un soporte mejorado para los elementos `<xsd:choice>`. Los asistentes generan un nuevo contenedor que almacena el valor asociado al elemento `<xsd:choice>`. Los asistentes generan estructuras de lenguaje que contienen un contenedor nuevo y un campo adicional:

fieldname -enum

El campo discriminante para indicar qué opciones de `<xsd:choice>` utilizará el elemento.

fieldname -cont

El nombre del contenedor que almacena la opción que se va a utilizar. Se genera una estructura de lenguaje adicional para correlacionar el valor de la opción.

El siguiente fragmento de esquema XML incluye un elemento `<xsd:choice>`:

```

<xsd:element name="choiceExample">
  <xsd:complexType>
    <xsd:choice>
      <xsd:element name="option1" type="xsd:string" />
      <xsd:element name="option2" type="xsd:int" />
      <xsd:element name="option3" type="xsd:short" maxOccurs="2" minOccurs="2" />
    </xsd:choice>
  </xsd:complexType>
</xsd:element>

```

Si este fragmento de esquema XML se procesa a un nivel de correlación 2.2 o superior, el asistente genera las siguientes estructuras de lenguaje COBOL:

```

03 choiceExample.
06 choiceExample-enum PIC X DISPLAY.
88 empty VALUE X'00'.
88 option1 VALUE X'01'.
88 option2 VALUE X'02'.
88 option3 VALUE X'03'.
06 choiceExample-cont PIC X(16).

01 Example-option1.
03 option1-length PIC S9999 COMP-5 SYNC.
03 option1 PIC X(255).

01 Example-option2.
03 option2 PIC S9(9) COMP-5 SYNC.

01 Example-option3.
03 option3 OCCURS 2 PIC S9999 COMP-5 SYNC.

```

Limitaciones para <xsd:choice> a nivel de correlación 2.2 y superior

DFHSC2LS y DFHWS2LS no soportan elemento <xsd:choice> anidados; por ejemplo, no se soporta el siguiente XML:

```

<xsd:choice>
  <xsd:element name="name1" type="string"/>
  <xsd:choice>
    <xsd:element name="name2a" type="string"/>
    <xsd:element name="name2b" type="string"/>
  </xsd:choice>
</xsd:choice>

```

DFHSC2LS y DFHWS2LS no soportan elementos <xsd:choice> recurrente; por ejemplo, no se soporta el siguiente XML:

```

<xsd:choice maxOccurs="2">
  <xsd:element name="name1" type="string"/>
</xsd:choice>

```

DFHSC2LS y DFHWS2LS soportan un máximo de 255 opciones en un elemento <xsd:choice>.

Soporte para <xsd:choice> a nivel de correlación 2.1 e inferior

En el nivel de correlación 2.1 e inferior, DFHWS2LS proporciona soporte limitado para elementos <xsd:choice>. DFHWS2LS trata cada una de las opciones del elemento <xsd:choice> aunque es un elemento <xsd:sequence> que se puede dar como máximo una vez.

Sólo se puede utilizar una de las opciones en un elemento <xsd:choice>, así que tenga cuidado cuando implemente una aplicación utilizando el elemento

<xsd:choice> ya que genera sólo combinaciones válidas de opciones. Cada uno de los elementos tiene su propio campo count en la estructura de lenguajes generada, exactamente uno de los cuales debe establecerse en 1 y los demás deben establecerse en 0. Cualquier otra combinación de valores es incorrecta, excepto cuando <xsd:choice> es opcional, en cuyo caso, es válido para todos los campos que se van a establecer en 0.

Soporte para los grupos de sustitución:

Puede utilizar un grupo de sustitución para definir un grupo de elementos XML que son intercambiables. Los asistentes de CICS proporcionan soporte para los grupos de sustitución a nivel de correlación 2.2 y superior.

A nivel de correlación 2.2 y superior, DFHSC2LS y DFHWS2LS soportan los grupos de sustitución utilizando correlaciones similares a las utilizadas para los elementos <xsd:choice>. El asistente genera un campo de enumeración y un nombre de contenedor nuevo en la estructura de lenguaje.

El siguiente fragmento de esquema XML incluye una matriz de dos elementos subGroupParent, cada uno de los cuales puede sustituirse por replacementOption1 o replacementOption2:

```
<xsd:element name="subGroupExample"
>
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="subGroupParent" maxOccurs="2" minOccurs="2" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:element name="subGroupParent" type="xsd:anySimpleType" />
<xsd:element name="replacementOption1" type="xsd:int"
substitutionGroup="subGroupParent" />
<xsd:element name="replacementOption2" type="xsd:short"
substitutionGroup="subGroupParent" />
```

Procesando este fragmento XML con el asistente genera las siguientes estructuras de lenguaje COBOL:

```
03 subGroupExample.
06 subGroupParent OCCURS2.
09 subGroupExample-enum PIC X DISPLAY.
88 empty VALUE X '00'.
88 replacementOption1 VALUE X '01'.
88 replacementOption2 VALUE X '02'.
88 subGroupParent VALUE X '03'.
09 subGroupExample-cont PIC X (16).
```

```
01 Example-replacementOption1.
03 replacementOption1 PIC S9(9) COMP-5 SYNC.
```

```
01 Example-replacementOption2.
03 replacementOption2 PIC S9999 COMP-5 SYNC.
```

```
01 Example-subGroupParent.
03 subGroupParent-length PIC S9999 COMP-5 SYNC.
03 subGroupParent PIC X(255).
```


Para obtener más información sobre los grupos de sustitución, consulte *W3C XML Schema Part 1: Structures Second Edition specification* : http://www.w3.org/TR/xmlschema-1/#Elements_Equivalence_Class

Soporte para elementos abstractos y tipos de datos abstractos:

Los asistentes de CICS proporcionan soporte para elementos abstractos y tipos de datos abstractos a nivel de correlación 2.2 y superior. Los asistentes de CICS correlacionan elementos abstractos y tipos de datos abstractos de forma similar a los grupos de sustitución.

Soporte para elementos abstractos a nivel de correlación 2.2 y superior

En el nivel de correlación 2.2 y superior, DFHSC2LS y DFHWS2LS tratan los elementos abstractos de casi la misma forma que los grupos de sustitución excepto que el elemento abstracto no es un miembro válido del grupo. Si no hay elementos sustituibles, el elemento abstracto se trata como un elemento `<xsd:any>` y utiliza las mismas correlaciones que el elemento `<xsd:any>` en el nivel de correlación 2.1.

El siguiente fragmento de esquema XML especifica dos opciones que se pueden utilizar en lugar del elemento abstracto. El elemento abstracto en sí mismo no es una opción válida:

```
<xsd:element name="abstractElementExample" >
<xsd:complexType>
<xsd:sequence>
<xsd:element ref="abstractElementParent" maxOccurs="2" minOccurs="2" />
</xsd:sequence>
</xsd:complexType>
</xsd:element>

<xsd:element name="abstractElementParent" type="xsd:anySimpleType"
abstract="true" />
<xsd:element name="replacementOption1" type="xsd:int"
substitutionGroup="abstractElementParent" />
<xsd:element name="replacementOption2" type="xsd:short"
substitutionGroup="abstractElementParent" />
```

Procesando este fragmento XML con el asistente genera las siguientes estructuras de lenguaje COBOL:

```
03 abstractElementExample.
06 abstractElementParent OCCURS 2.
09 abstractElementExample-enum PIC X DISPLAY.
88 empty VALUE X '00'.
88 replacementOption1 VALUE X '01'.
88 replacementOption2 VALUE X '02'.
09 abstractElementExample-cont PIC X (16).
```

```
01 Example-replacementOption1.
03 replacementOption1 PIC S9(9) COMP-5 SYNC.
```

```
01 Example-replacementOption2.
03 replacementOption2 PIC S9999 COMP-5 SYNC.
```

Para obtener más información sobre los elementos abstractos, consulte *W3C XML Schema Part 0: Primer Second Edition specification* : <http://www.w3.org/TR/xmlschema-0/#SubsGroups>

Soporte para tipos de elementos abstractos a nivel de correlación 2.2 y superior

En el nivel de correlación 2.2 y superior, DFHSC2LS y DFHWS2LS tratan los tipos de datos abstractos como grupos de sustitución. El asistente genera un campo de enumeración y un nombre de contenedor nuevo en la estructura de lenguaje.

El siguiente fragmento de esquema XML especifica dos alternativas que se pueden utilizar en lugar del tipo abstracto:

```
<xsd:element name="AbstractDataTypeExample"
type="abstractDataType" />

<xsd:complexType name="abstractDataType" abstract="true">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string" />
  </xsd:simpleContent>
</xsd:complexType>
<xsd:complexType name="option1">
  <xsd:simpleContent>
    <xsd:restriction base="abstractDataType">
      <xsd:length value="5" />
    </xsd:restriction>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:complexType name="option2">
  <xsd:simpleContent>
    <xsd:restriction base="abstractDataType">
      <xsd:length value="10" />
    </xsd:restriction>
  </xsd:simpleContent>
</xsd:complexType>
```

Procesando este fragmento XML con el asistente genera las siguientes estructuras de lenguaje COBOL:

```
03 AbstractDataTypeExamp-enum PIC X DISPLAY.
88 empty VALUE X'00'.
88 option1 VALUE X'01'.
88 option2 VALUE X'02'.
03 AbstractDataTypeExamp-cont PIC X(16).
```

Las estructuras de lenguaje se generan en libros de copias individuales. La estructura de lenguaje generada para option1 se genera en un libro de copias:

```
03 option1 PIC X(5).
```

La estructura de lenguaje para option2 se genera en un libro de copias diferente:

```
03 option2 PIC X(10).
```

Para obtener más información sobre los tipos de datos abstractos, consulte *W3C XML Schema Part 0: Primer Second Edition specification* : <http://www.w3.org/TR/xmlschema-0/#SubsGroups>

Cómo gestionar el contenido que se repite variablemente en COBOL:

En COBOL, no puede procesar el contenido que se repite variablemente utilizando aritmética de punteo para dirigir cada instancia de datos. Otros lenguajes de programación no tienen esta limitación. Este ejemplo le muestra cómo gestionar el contenido que se repite variablemente en COBOL para una aplicación de servicio web.

Esta técnica también se aplica para transformar XML en datos de aplicación utilizando los mandatos de API **TRANSFORM**. Ek siguiente documento WSDL de

ejemplo representa un servicio web con datos de aplicación que constan de una serie de 8 caracteres que recorre un número de veces variable:

```
<?xml version="1.0"?>
  <definitions name="ExampleWSDL"
    targetNamespace="http://www.example.org/variablyRepeatingData/"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:tns="http://www.example.org/variablyRepeatingData/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">

    <types>
      <xsd:schema targetNamespace="http://www.example.org/variablyRepeatingData/">
        <xsd:element name="applicationData">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="component" minOccurs="1" maxOccurs="unbounded">
                <xsd:simpleType>
                  <xsd:restriction base="xsd:string">
                    <xsd:length value="8"/>
                  </xsd:restriction>
                </xsd:simpleType>
              </xsd:element>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:schema>
    </types>

    <message name="exampleMessage">
      <part element="tns:applicationData" name="messagePart"/>
    </message>

    <portType name="examplePortType">
      <operation name="exampleOperation">
        <input message="tns:exampleMessage"/>
        <output message="tns:exampleMessage"/>
      </operation>
    </portType>

    <binding name="exampleBinding" type="tns:examplePortType">
      <soap:binding style="document"
        transport="http://schemas.xmlsoap.org/soap/http"/>
      <operation name="exampleOperation">
        <soap:operation soapAction=""/>
        <input><soap:body parts="messagePart" encodingStyle="" use="literal"/></input>
        <output><soap:body parts="messagePart" encodingStyle=""
          use="literal"/></output>
      </operation>
    </binding>
  </definitions>
```

El proceso de este documento WSDL a través de DFHWS2LS genera las siguientes estructuras de lenguaje COBOL:

```
03 applicationData.
```

```
06 component-num PIC S9(9) COMP-5 SYNC.
06 component-cont PIC X(16).
```

```
01 DFHWS-component.
03 component PIC X(8).
```

Tenga en cuenta que el campo component de 8 caracteres se define en una estructura individual llamada DFHWS-component. La estructura de datos principal se denomina applicationData y contiene dos campos, component-num y component-cont. El campo component-num indica cuántas instancias de datos

component están presentes y el campo component-cont indica el nombre de un contenedor que contiene la lista concatenada de campos component.

El siguiente código COBOL muestra una forma de procesar la lista de datos recurrentes de forma variable. Utiliza una matriz de Linkage Section para direccionar instancias posteriores de datos, cada uno de los cuales se visualiza utilizando la sentencia DISPLAY:

```
IDENTIFICATION DIVISION.  
PROGRAM-ID. EXVARY.
```

```
ENVIRONMENT DIVISION.  
DATA DIVISION.  
WORKING-STORAGE SECTION.
```

```
* variables de almacenamiento de trabajo  
01 APP-DATA-PTR USAGE IS POINTER.  
01 APP-DATA-LENGTH PIC S9(8) COMP.  
01 COMPONENT-PTR USAGE IS POINTER.  
01 COMPONENT-DATA-LENGTH PIC S9(8) COMP.  
01 COMPONENT-COUNT PIC S9(8) COMP-4 VALUE 0.  
01 COMPONENT-LENGTH PIC S9(8) COMP.
```

```
LINKAGE SECTION.
```

```
* una matriz de linkage section grande  
01 BIG-ARRAY PIC X(659999).
```

```
* estructuras de datos de aplicación creadas por DFHWS2LS  
* normalmente se hace referencia a esto con una sentencia COPY  
01 DFHWS2LS-data.  
03 applicationData.  
06 component-num PIC S9(9) COMP-5 SYNC.  
06 component-cont PIC X(16).
```

```
01 DFHWS-component.  
03 component PIC X(8).
```

```
PROCEDURE DIVISION USING DFHEIBLK.  
A-CONTROL SECTION.  
A010-CONTROL.
```

```
* Obtener contenedor DFHWS-DATA  
EXEC CICS GET CONTAINER('DFHWS-DATA')  
SET(APP-DATA-PTR)  
FLENGTH(APP-DATA-LENGTH)  
END-EXEC  
SET ADDRESS OF DFHWS2LS-data TO APP-DATA-PTR
```

```
* Obtener datos de componente recurrentes  
EXEC CICS GET CONTAINER(component-cont)  
SET(COMPONENT-PTR)  
FLENGTH(COMPONENT-DATA-LENGTH)  
END-EXEC
```

```
* Señalar la estructura de componente en la primera instancia de  
los datos  
SET ADDRESS OF DFHWS-component TO COMPONENT-PTR
```

```
* Almacenar la longitud de un único componente  
MOVE LENGTH OF DFHWS-component TO COMPONENT-LENGTH
```

```
* procesar cada instancia de datos de componente de una en una  
PERFORM WITH TEST AFTER
```

```

UNTIL COMPONENT-COUNT = component-num

* visualizar la instancia actual de los datos
DISPLAY 'component value is: ' component

* direccionar la siguiente instancia de datos de componente
SET ADDRESS OF BIG-ARRAY TO ADDRESS OF DFHWS-component
SET ADDRESS OF DFHWS-component
TO ADDRESS OF BIG-ARRAY (COMPONENT-LENGTH + 1:1)
ADD 1 TO COMPONENT-COUNT

* finalizar el bucle
END-PERFORM.

* Señalar la estructura de componente de vuelta a la primera instancia de
* los datos, para cualquier proceso posterior que deseemos llevar a cabo
SET ADDRESS OF DFHWS-component TO COMPONENT-PTR

* volver a CICS.

EXEC CICS
RETURN
END-EXEC

GOBACK.

```

El código anterior proporciona una solución genérica para manejar contenido que se repite de forma variable. La matriz, BIG-ARRAY, se mueve al inicio de cada componente a su vez y no permanece fijo al inicio de los datos. La estructura de datos de componente se mueve para señalar el primer byte del siguiente componente. COMPONENT-PTR puede utilizarse para recuperar la posición de inicio de los datos de componente si es necesario.

Aquí tiene un mensaje SOAP de ejemplo que se ajusta al documento WSDL:

```

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body>
<applicationData xmlns="http://www.example.org/variablyRepeatingData/">
<component xmlns="">VALUE1</component>
<component xmlns="">VALUE2</component>
<component xmlns="">VALUE3</component>
</applicationData>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

Aquí tiene la salida producida por el programa COBOL cuando procesa el mensaje SOAP:

```

CPIH 20080115103151 component value is: VALUE1
CPIH 20080115103151 component value is: VALUE2
CPIH 20080115103151 component value is: VALUE3

```

Matrices de variables de elementos

XML puede contener una matriz con distintos números de elementos. En general, los documentos WSDL y esquemas XML que contienen varios números de elementos no se correlacionan de forma eficiente en una única estructura de datos de lenguaje de alto nivel. CICS utiliza correlaciones basadas en contenedores o correlaciones en línea para gestionar varios elementos en XML.

Una matriz con varios elementos se representa en el esquema XML utilizando los atributos minOccurs y maxOccurs en la declaración del elemento:

- El atributo `minOccurs` especifica el número de veces mínimo que se puede dar el elemento. Puede tener un valor de 0 o un entero positivo.
- El atributo `maxOccurs` especifica el número de veces máximo que se puede dar el elemento. Puede tener un valor de un entero positivo mayor o igual al valor del atributo `minOccurs`. También puede tomar un valor de `unbounded`, que indica que no se aplica límite superior al número de veces que se puede dar el elemento.
- El valor predeterminado para ambos atributos es 1.

Este ejemplo indica una serie de 8 bytes que es opcional; es decir, que puede que no ocurra nunca o que ocurra una vez en el mensaje SOAP o XML de aplicación:

```
<xsd:element name="component"
minOccurs="0" maxOccurs="1">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:length value="8"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

El ejemplo siguiente indica una serie de 8 bytes que debe tener lugar al menos una vez:

```
<xsd:element name="component"
minOccurs="1" maxOccurs="unbounded">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:length value="8"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

En general, los documentos WSDL que contienen varios números de elementos no se correlacionan de forma eficiente en una única estructura de datos de lenguaje de alto nivel. Por lo tanto, para gestionar estos casos, CICS utiliza una serie de estructuras de datos conectadas que se pasan al programa de aplicación en una serie de contenedores. Estas estructuras se utilizan como entrada y salida de la aplicación:

- Cuando CICS transforma XML en datos de aplicación, llena estas estructuras con los datos de aplicación y la aplicación los lee.
- Cuando CICS transforma los datos de aplicación en XML, lee los datos de aplicación en las estructuras que ha estado llenando la aplicación.

El formato de estas estructuras de datos se explica mejor con una serie de ejemplos. El XML puede ser de un mensaje SOAP o de una aplicación. Estos ejemplos utilizan una matriz de campos simples de 8 bytes. Sin embargo, el modelo soporta matrices de tipos de datos complejos y matrices de tipos de datos que contienen otras matrices.

Número fijo de elementos

El primer ejemplo ilustra un elemento que se produce tres veces exactamente:

```
<xsd:element name="component"
minOccurs="3" maxOccurs="3">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
```

```

<xsd:length value="8"/>
</xsd:restriction>
</xsd:simpleType>
</xsd:element>

```

En este ejemplo, debido a que el número de veces que se da un elemento se conoce de antemano, se puede representar como una matriz de longitud fija en una sencilla declaración COBOL (o el equivalente en otros lenguajes):

```
05 component PIC X(8) OCCURS 3 TIMES
```

Número variado de elementos en el nivel de correlación 2 e inferior

Este ejemplo ilustra un elemento obligatorio que puede encontrarse de una a cinco veces:

```

<xsd:element name="component"
minOccurs="1" maxOccurs="5">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:length value="8"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>

```

La estructura de datos principal contiene una declaración de dos campos. Cuando CICS transforma el XML en datos binarios, el primer campo component-num contiene el número de veces que aparece el elemento en el XML, y el segundo campo, component-cont, contiene el nombre del contenedor:

```
05 component-num PIC S9(9) COMP-5
05 component-cont PIC X(16)
```

Una segunda estructura de datos contiene la declaración del elemento:

```
01 DFHWS-component
02 component PIC X(8)
```

Debe examinar el valor de component-num (que contendrá un valor en el rango de 1 a 5) para averiguar cuántas veces se da el elemento. El contenido del elemento está en el contenedor denominado component-cont; el contenedor tiene una matriz de elementos, donde cada elemento está correlacionado por la estructura de datos DFHWS-component.

Si minOccurs="0" y maxOccurs="1", el elemento es opcional. Para procesar la estructura de datos en el programa de aplicación, debe examinar el valor de component-num:

- Si es cero, el mensaje no tiene elemento de componente y el contenido de component-cont no está definido.
- Si es uno, el elemento de componente está en el contenedor denominado component-cont.

El contenido del contenedor está correlacionado por la estructura de datos DFHWS-component.

Nota: Si el mensaje SOAP consta de un único elemento recurrente, DFHWS2LS genera dos estructuras de lenguaje. La estructura de lenguaje principal contiene el número de elementos en la matriz y el nombre del contenedor que tiene la matriz de elementos. La segunda estructura de lenguaje correlaciona una única instancia de elemento recurrente.

Número variado de elementos en el nivel de correlación 2.1 y superior

En el nivel de correlación 2.1 y superior, puede utilizar el parámetro **INLINE-MAXOCCURS-LIMIT** en los asistentes de CICS. El parámetro **INLINE-MAXOCCURS-LIMIT** especifica la forma en la que se manejan varios elementos. Las opciones de correlación para un número variado de elementos son correlaciones basadas en contenedores, descritas en “Número variado de elementos en el nivel de correlación 2 e inferior” en la página 350, o correlaciones en línea. El *valor* de este parámetro puede ser un entero positivo en el rango de 0 a 32767:

- El valor predeterminado de **INLINE-MAXOCCURS-LIMIT** es 1, que garantiza que los elementos opcionales están correlacionados en línea.
- Un valor de 0 para el parámetro **INLINE-MAXOCCURS-LIMIT** impide la correlación en línea.
- Si maxOccurs es menor o igual al valor de **INLINE-MAXOCCURS-LIMIT**, se utiliza la correlación en línea.
- Si maxOccurs es mayor que el valor de **INLINE-MAXOCCURS-LIMIT**, se utiliza la correlación basada en contenedores.

La correlación de un número variado de elementos en línea da como resultado la generación de una matriz, como sucede con el ejemplo de aparición fija anterior, y un contador. El campo component-num indica cuántas instancias del elemento están presentes y si están señaladas por una matriz. Para el ejemplos mostrado en “Número variado de elementos en el nivel de correlación 2 e inferior” en la página 350, cuando **INLINE-MAXOCCURS-LIMIT** es menor o igual a 5, la estructura de datos generada es como ésta:

```
05 component-num PIC S9(9) COMP-5 SYNC.  
05 component OCCURS 5 PIC X(8).
```

El primer campo, component-num, es idéntico a la salida para el ejemplo de correlación basada en contenedores de la sección anterior. El segundo campo contiene una matriz de longitud 5 que es lo suficientemente larga para contener el número máximo de elementos que puede generar.

La correlación en línea difiere de la correlación basada en contenedores, que almacena el número de apariciones del elemento y el nombre del contenedor donde están colocados los datos, porque almacena todos los datos en el contenedor actual. Almacenar los datos en el contenedor actual normalmente mejorará el rendimiento y hará que la correlación en línea sea preferible.

Matrices de variables anidadas

Los esquemas XML y documentos WSDL complejos pueden contener elementos recurrentes variables, que a su vez contienen elementos recurrentes variables. En este caso, la estructura descrita se amplía más allá de los dos niveles descritos en los ejemplos.

Este ejemplo ilustra en elemento opcional denominado <component2> que se anida en un elemento obligatorio denominado <component1>, donde el elemento obligatorio puede encontrarse de una a cinco veces:

```
<xsd:element name="component1"  
  minOccurs="1" maxOccurs="5">  
  <xsd:complexType>  
    <xsd:sequence>  
      <xsd:element name="component2"  
        minOccurs="0" maxOccurs="1">  
        <xsd:simpleType>
```



```

<xsd:restriction base="xsd:string">
<xsd:length value="8"/>
</xsd:restriction>
</xsd:simpleType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>

```

La estructura de datos de nivel superior es exactamente la misma que en los ejemplos anteriores:

```

05 component1-num PIC S9(9) COMP-5
05 component1-cont PIC X(16)

```

Sin embargo, la segunda estructura de datos contiene estos elementos:

```

01 DFHWS-component1
02 component2-num PIC S9(9) COMP-5
02 component2-cont PIC X(16)

```

Una estructura de tercer nivel contiene estos elementos:

```

01 DFHWS-component2
02 component2 PIC X(8)

```

El número de apariciones del elemento más externo "component1" está en component1-num.

El contenedor nombrado en component1-cont contiene una matriz con varias instancias de la segunda estructura de datos DFHWS-component1.

Cada instancia de component2-cont nombra un contenedor diferente, cada uno de los cuales contiene la estructura de datos correlacionada por la estructura de tercer nivel DFHWS-component2.

Para ilustrar esta estructura, tenga en cuenta el fragmento de XML que coincide con el ejemplo:

```

<component1><component2>
string1
</component2></component1>
<component1><component2>
string2
</component2></component1>
<component1></component1>

```

<component1> se da tres veces. Las dos primeras contienen cada una una instancia de <component2>; la tercera instancia no.

En la estructura de datos de nivel superior, component1-num contiene un valor de 3. El contenedor nombrado en component1-cont tiene tres instancias de DFHWS-component1 :

1. En la primera, component2-num tiene un valor de 1, y el contenedor nombrado en component2-cont contiene *string1*.
2. En la segunda, component2-num tiene un valor de 1, y el contenedor nombrado en component2-cont contiene *string2*.
3. En la tercera, component2-num tiene un valor de 0, y el contenido de component2-cont no está definido.

En esta instancia, la estructura de datos completa se representa por medio de cuatro contenedores en total:

- La estructura de datos raíz en el contenedor DFHWS-DATA
- El contenedor nombrado en component1-cont
- Los dos contenedores nombrados en las dos primeras instancias de component2-cont.

Estructuras opcionales y xsd:choice

DFHWS2LS y DFHSC2LS soportan el uso de maxOccurs y minOccurs en elementos <xsd:sequence>, <xsd:choice>, y <xsd:all> sólo en el nivel de correlación 2.1 y superior, donde los atributos minOccurs y maxOccurs están configurados en minOccurs="0" y maxOccurs="1".

Los asistentes generan correlaciones que tratan estos elementos como si cada elemento hijo en ellos fuera opcional. Cuando implementa una aplicación con estos elementos, asegúrese de que la aplicación no genera combinaciones no válidas de opciones. Cada uno de los elementos tiene su propio campo count en la estructura de lenguajes generada, estos campos deben establecerse todos en "0" o todos en "1". Cualquier otra combinación de valores no es válida; excepto con elementos <xsd:choice>.

Los elementos <xsd:choice> indican que sólo se puede utilizar una de las opciones del elemento. Esto se soporta en todos los niveles de correlación. Los asistentes gestionan cada una de las opciones en un <xsd:choice> aunque es un elemento <xsd:sequence> con minOccurs="0" y maxOccurs="1". Tenga cuidado cuando implementa una aplicación utilizando el elemento <xsd:choice> para asegurarse de que la aplicación no genera combinaciones no válidas de las opciones. Cada uno de los elementos tiene su propio campo count en la estructura de lenguajes generada, exactamente uno debe estar establecido en '1' y los demás establecidos en '0'. Cualquier otra combinación de valores no es válida, excepto cuando el elemento <xsd:choice> es opcional, en cuyo caso es válido para todos los campos que esté establecido en '0'.

Soporte para valores de longitud variable y espacio en blanco

Puede personalizar la forma en que se gestionan los valores de longitud variable y el espacio en blanco utilizando valores en los asistentes de CICS y añadiendo facetas directamente en el esquema XML.

Normalmente, el asistente XML de CICS y el asistente de servicios web de CICS correlacionan series de datos con matrices de caracteres de longitud fija; estas matrices requieren relleno con espacios o nulos. La correlación de valores de longitud variables a matrices de datos de longitud fija puede ser ineficaz y un almacenamiento de residuos. Si la longitud de datos es variable, se recomienda personalizar la manera en la que se manejan estas correlaciones.

Si está convirtiendo desde una estructura de lenguaje a un esquema XML o documento WSDL, se le recomienda especificar las facetas whiteSpace y maxLength en el esquema XML y configurar el parámetro **CHAR-VARYING-LIMIT** en los asistentes.

Si está convirtiendo desde un esquema XML o documento WSDL a una estructura de lenguaje, se recomienda establecer un valor adecuado para el parámetro **CHAR-VARYING** en los asistentes.

Nota: Los caracteres nulos ('x00') no son válidos en documentos XML. Los caracteres nulo de los datos de aplicación analizados por CICS se consideran para indicar el final de la serie y el valor se trunca. Cuando CICS genera datos de

aplicación, lo hace según el valor del parámetro **CHAR-VARYING**. Por ejemplo, si se especifica la opción **CHAR-VARYING=NULL**, las series de longitud variable generadas por CICS terminan con un carácter nulo.

Correlación de valores de longitud variable desde XML a estructuras de lenguaje

Utilice facetas en el esquema XML o especifique determinados parámetros en los asistentes de CICS para personalizar la forma en la que se manejan las correlaciones entre el esquema XML o el documento WSDL y la estructura de lenguaje.

Los tipos de datos XML se pueden restringir utilizando facetas. Utilice las facetas de longitud (`length` , `maxLength` y `minLength`) y la faceta `whiteSpace` para personalizar cómo se manejan los datos de longitud variables en el XML.

length

Se utiliza para especificar si los datos son de longitud fija.

maxLength

Se utiliza para especificar la longitud máxima para el tipo de datos. Si no se establece este valor para un tipo de datos basado en series, la longitud máxima no está limitada.

minLength

Se utiliza para especificar la longitud mínima para el tipo de datos. Si no se establece este valor para un tipo de datos basado en series, la longitud mínima es 0.

whiteSpace

Se utiliza para especificar cómo se maneja el espacio en blanco alrededor de un valor de datos. Los espacios en blanco incluyen espacios, pestañas y líneas nuevas. La faceta `whiteSpace` se puede establecer en `preserve` , `replace` o `collapse` :

- Un valor de `preserve` mantiene cualquier espacio en blanco en los valores de datos.
- Un valor de `replace` significa que cualquier pestaña o línea nueva se sustituye con el número de espacios adecuados.
- Un valor de `collapse` significa que los en blanco iniciales, finales e incluidos se eliminan, y todas las pestañas, líneas nuevas y espacios consecutivos se sustituyen por caracteres de un sólo espacio.

Si la faceta `whiteSpace` no está establecida, se conserva el espacio en blanco.

Para obtener más información sobre las facetas de esquema XML, consulte el esquema de recomendación W3C *XML Schema Part 2: Datatypes Second Edition* <http://www.w3.org/TR/xmlschema-2/#facets>

Se pueden utilizar los parámetros siguientes en los asistentes de CICS, DFHSC2LS y DFHWS2LS, para alterar la forma en la que los datos de longitud variable se correlacionan desde el esquema XML a la estructura de lenguaje. Estos parámetros están disponibles en el nivel de correlación 1.2 o superior.

DEFAULT-CHAR-MAXLENGTH

Especifica la longitud de matriz predeterminada de los datos de caracteres en caracteres para correlaciones donde no está implicada la longitud en el

esquema XML o el documento WSDL. El valor de este parámetro puede ser un entero positivo en el rango de 1 a 2 147 483 647.

Sin embargo, se recomienda especificar la longitud de caracteres máxima que desea que utilicen DFHSC2LS o DFHWS2LS directamente en el esquema XML o documento WSDL con la faceta `maxLength`. Especificando la longitud máxima directamente en el esquema XML o documento WSDL evita problemas asociados con tener un valor predeterminado global aplicado a todos los tipos de datos basados en series.

CHAR-VARYING-LIMIT

Especifica el tamaño máximo de los datos de caracteres de longitud variable que se correlacionan con la estructura de lenguaje. Si los datos de caracteres son mayores que el valor especificado en este parámetro, se correlacionan con un contenedor y el nombre de contenedor se utiliza en la estructura de lenguaje generada. El valor puede estar en el rango de 0 al valor predeterminado 32 767 bytes.

CHAR-VARYING

Especifica cómo se correlacionan los datos de caracteres de longitud variable. Si no especifica este parámetro, la correlación predeterminada depende del lenguaje especificado. Puede seleccionar estas opciones:

- **CHAR-VARYING=NO** especifica que los datos de caracteres de longitud variable se correlacionan como series de longitud fija.
- **CHAR-VARYING=NULL** especifica que los datos de caracteres de longitud variable se correlacionan con series terminadas en nulo.
- **CHAR-VARYING=YES** especifica que los datos de caracteres de longitud variable se correlacionan con un tipo de datos CHAR VARYING en PL/I. En lenguajes COBOL, C y C++, los datos de caracteres de longitud variable se correlacionan con una representación equivalente que consta de dos elementos relacionados: la longitud de datos y los datos.

El valor **CHAR-VARYING=YES** normalmente da como resultado el mejor rendimiento.

Correlación de valores de longitud variable desde estructuras de lenguaje a XML

Puede personalizar la forma en la que se manejan las correlaciones entre la estructura de lenguaje y el esquema XML o documento WSDL. Establezca el parámetro **CHAR-VARYING** en DFHLS2SC o DFHLS2WS, en **COLLAPSE** o **NULL** para cambiar la manera en la que se generan las matrices de caracteres.

Establezca la opción **CHAR-VARYING=NULL** que le dice a CICS que añada un carácter nulo al final de cada matriz de caracteres cuando genera XML.

Establezca la opción **CHAR-VARYING=COLLAPSE** que le dice a CICS que elimine automáticamente cualquier espacio final del final de las matrices de caracteres al generar XML. Esta opción está disponible sólo en los niveles de correlación 2.1 o superior y **CHAR-VARYING=COLLAPSE** es el valor predeterminado en el nivel de correlación 2.1 o superior para todos los lenguajes distintos a C y C++. Cuando se analiza el XML, se eliminan todos los espacios en blanco de inicio, final e incluidos.

Para obtener más información, consulte Soporte para valores de longitud variable y espacios en blanco en servicios web de CICS (nota técnica) .

Matrices de variables de elementos en DFHJS2LS

JSON puede incluir matrices de distintos números de elementos. En general, los esquemas JSON que contienen varios números de elementos no se correlacionan de forma eficiente en una única estructura de datos de lenguaje de alto nivel. CICS utiliza correlaciones basadas en contenedores o correlaciones en línea para gestionar varios elementos en datos JSON.

Una matriz con varios elementos se representa en el esquema JSON utilizando las palabras clave `minItems` y `maxItems` en el esquema con el valor `"type"` de `"array"` :

- La palabra clave `minItems` especifica el número de veces mínimo que se puede dar un elemento. Puede tener un valor de 0 o un entero positivo. El valor predeterminado es 0.
- La palabra clave `maxItems` especifica el número de veces máximo que se puede dar un elemento. Puede tener un valor de entero positivo mayor o igual al valor de la palabra clave `minItems`.
- Si falta la palabra clave `maxItems`, significa que la matriz no está limitada.

Se puede indicar un campo opcional por medio de una matriz de variables `"maxItems":1` . Por ejemplo, una serie de 8 bytes opcional denominada `"component"` :

```
"properties":{
  "component": {
    "type":"array",
    "maxItems":1,
    "items": {
      "type": "string",
      "maxLength": 8
    }
  },
  "required": ["component"]
}
```

Se puede obtener el mismo efecto no incluyendo el nombre de campo en el valor de palabra clave `"required"`:

```
"properties":{
  "component": {
    "type": "string",
    "maxLength": 8
  }
}
```

En general, los esquemas JSON que contienen varios números de elementos no se correlacionan de forma eficiente en una única estructura de datos de lenguaje de alto nivel. Para gestionar estos casos, CICS utiliza una serie de estructuras de datos conectadas que se pasan al programa de aplicación en una serie de contenedores. Estas estructuras se utilizan como entrada y salida de la aplicación:

- Cuando CICS transforma datos JSON en datos de aplicación, llena estas estructuras con los datos de aplicación y la aplicación los lee.
- Cuando CICS transforma los datos de aplicación en datos JSON, lee los datos de aplicación en las estructuras que ha estado llenando la aplicación.

Los ejemplos siguientes ilustran el formato de estas estructuras de datos. Estos ejemplos utilizan una matriz de campos simples de 8 bytes. Sin embargo, el modelo soporta matrices de tipos de datos complejos y matrices de tipos de datos que contienen otras matrices.

Ejemplo 1. Número fijo de elementos

Este ejemplo ilustra un elemento que se produce tres veces exactamente:

```
"properties":{
  "component": {
    "type": "array",
    "maxItems": 3,
    "minItems": 3,
    "items": {
      "type": "string",
      "maxLength": 8
    }
  },
  "required": ["component"]
}
```

En este ejemplo, el número de veces que se da un elemento se conoce de antemano, por lo tanto, se puede representar como una matriz de longitud fija en una sencilla declaración COBOL o el equivalente en otros lenguajes.

```
05 component PIC X(8) OCCURS 3 TIMES
```

Ejemplo 2. Número de elementos variable en el nivel de correlación 2 e inferior

Este ejemplo ilustra un elemento obligatorio que puede encontrarse de una a cinco veces:

```
"properties":{
  "component": {
    "type": "array",
    "maxItems": 5,
    "minItems": 1,
    "items": {
      "type": "string",
      "maxLength": 8
    }
  },
  "required": ["component"]
}
```

La estructura de datos principal contiene una declaración de dos campos. Cuando CICS transforma los datos JSON en datos binarios, el primer campo component-num contiene el número de veces que aparece el elemento en los datos JSON, y el segundo campo, component-cont , contiene el nombre de un contenedor:

```
05 component-num PIC S9(9) COMP-5
05 component-cont PIC X(16)
```

Una segunda estructura de datos contiene la declaración del elemento:

```
01 DFHJS-component
02 component PIC X(8)
```

Debe examinar el valor de component-num , que contendrá un valor en el rango de 1 a 5, para averiguar cuántas veces se da el elemento. El contenido del elemento está en el contenedor denominado component-cont; el contenedor tiene una matriz de elementos, donde cada elemento está correlacionado por la estructura de datos DFHJS-component.

Si minItems="0" o falta y maxItems="1" , el elemento es opcional. Para procesar la estructura de datos en el programa de aplicación, debe examinar el valor de component-num:

- Si es cero, el mensaje no tiene elemento de componente y el contenido de `component-cont` no está definido.
- Si es uno, el elemento de componente está en el contenedor denominado `component-cont`.

El contenido del contenedor está correlacionado por la estructura de datos `DFHJS-component`.

Nota: Si los datos JSON constan de un único elemento recurrente, `DFHJS2LS` genera dos estructuras de lenguaje. La estructura de lenguaje principal contiene el número de elementos en la matriz y el nombre del contenedor que tiene la matriz de elementos. La segunda estructura de lenguaje correlaciona una única instancia de elemento recurrente.

Ejemplo 3. Número variable de elementos a nivel de correlación 2.1 y superior

En el nivel de correlación 2.1 y superior, puede utilizar el parámetro **INLINE-MAXOCCURS-LIMIT** en los asistentes de CICS. El parámetro **INLINE-MAXOCCURS-LIMIT** especifica la forma en la que se manejan varios elementos. Las opciones de correlación para un número variado de elementos son correlaciones basadas en contenedores, descritas en “Ejemplo 2. Número de elementos variable en el nivel de correlación 2 e inferior” en la página 285, o correlaciones en línea. El valor de este parámetro puede ser un entero positivo en el rango de 0 a 32767:

- El valor predeterminado de **INLINE-MAXOCCURS-LIMIT** es 1, que garantiza que los elementos opcionales están correlacionados en línea.
- Un valor de 0 para el parámetro **INLINE-MAXOCCURS-LIMIT** impide la correlación en línea.
- Si `maxItems` es menor o igual al valor de **INLINE-MAXOCCURS-LIMIT**, se utiliza la correlación en línea.
- Si `maxItems` es mayor que el valor de **INLINE-MAXOCCURS-LIMIT**, se utiliza la correlación basada en contenedores.

La correlación de un número variado de elementos en línea da como resultado la generación de una matriz, como sucede con el ejemplo de aparición fija anterior, y un contador. El campo `component-num` indica cuántas instancias del elemento están presentes y si están señaladas por una matriz. Para el ejemplos mostrado en “Ejemplo 2. Número de elementos variable en el nivel de correlación 2 e inferior” en la página 285, cuando **INLINE-MAXOCCURS-LIMIT** es menor o igual a 5, la estructura de datos generada es como ésta:

```
05 component-num PIC S9(9) COMP-5 SYNC.
05 component OCCURS 5 PIC X(8).
```

El primer campo, `component-num`, es idéntico a la salida para el ejemplo de correlación basada en contenedores de la sección anterior. El segundo campo contiene una matriz de longitud 5 que es lo suficientemente larga para contener el número máximo de elementos que puede generar.

La correlación en línea difiere de la correlación basada en contenedores, que almacena el número de apariciones del elemento y el nombre del contenedor donde están colocados los datos, porque almacena todos los datos en el contenedor actual. Almacenar los datos en el contenedor actual normalmente mejorará el rendimiento y hará que la correlación en línea sea preferible.

Ejemplo 4. Matrices de variables anidadas

Los esquemas JSON complejos pueden contener elementos recurrentes variables, que a su vez contienen elementos recurrentes variables. En este caso, la estructura descrita se amplía más allá de los dos niveles descritos en los ejemplos.

Este ejemplo ilustra un elemento opcional llamado "component2" que se anida en un elemento obligatorio llamado "component1" , donde el elemento obligatorio se puede dar de una a cinco veces:

```
"properties":{
  "component1": {
    "type": "array",
    "maxItems": 5,
    "minItems": 1,
    "items": {
      "type": "object",
      "properties":{
        "component2":{
          "type": "string",
          "maxLength": 8
        }
      },
      "required": ["component2"]
    }
  },
  "required": ["component1"]
}
```

La estructura de datos de nivel superior es exactamente la misma que en los ejemplos anteriores:

```
05 component1-num PIC S9(9) COMP-5
05 component1-cont PIC X(16)
```

Sin embargo, la segunda estructura de datos contiene estos elementos:

```
01 DFHJS-component1
02 component2-num PIC S9(9) COMP-5
02 component2-cont PIC X(16)
```

Una estructura de tercer nivel contiene estos elementos:

```
01 DFHJS-component2
02 component2 PIC X(8)
```

El número de apariciones del elemento más externo "component1" está en component1-num.

El contenedor nombrado en component1-cont contiene una matriz con varias instancias de la segunda estructura de datos DFHJS-component1.

Cada instancia de component2-cont nombra un contenedor diferente, cada uno de los cuales contiene la estructura de datos correlacionada por la estructura de tercer nivel DFHJS-component2.

Para ilustrar esta estructura, tenga en cuenta el fragmento de datos JSON que coincide con el ejemplo:

```
{"component1":
[
{
  "component2": "string1"
},
{

```



```
"component2": "string2"
},
]
```

"component1" se da tres veces. Las dos primeras contienen una instancia de "component2" ; la tercera instancia no.

En la estructura de datos de nivel superior, component1-num contiene un valor de 3. El contenedor nombrado en component1-cont tiene tres instancias de DFHJS-component1 :

1. En la primera, component2-num tiene un valor de 1, y el contenedor nombrado en component2-cont contiene *string1*.
2. En la segunda, component2-num tiene un valor de 1, y el contenedor nombrado en component2-cont contiene *string2*.
3. En la tercera, component2-num tiene un valor de 0, y el contenido de component2-cont no está definido.

En esta instancia, la estructura de datos completa se representa por medio de cuatro contenedores en total:

- La estructura de datos raíz en el contenedor DFHJS-DATA.
- El contenedor nombrado en component1-cont.
- Los dos contenedores nombrados en las dos primeras instancias de component2-cont.

Estructuras opcionales y la palabra clave required

Las estructuras de datos se definen por el esquema JSON "type" de "object". Los esquemas relacionan nombres de campos con tipos individuales utilizando el objeto proporcionado por la palabra clave "properties". El requisito para que estos campos sean parte de los datos JSON descritos por el esquema JSON se controla por medio de la matriz proporcionada por la palabra clave "required". Esta matriz lista todos los nombres de campo que deben estar presentes en los datos JSON. Por lo tanto, los campos opcionales están representados por su ausencia de la matriz, o cuando la matriz no puede estar vacía, la ausencia de la palabra clave "required". En este caso, todos los campos son opcionales.

Los campos opcionales se tratan como matriz variable de 0 o 1 elementos. Esto añade un campo adicional con el sufijo "-num" añadido al nombre de elemento. Si la longitud total es mayor de 28 caracteres, el nombre de elemento se trunca. En el tiempo de ejecución, no será cero para indicar que el valor estaba presente en los datos JSON y cero si el valor no lo está.

Este ejemplo muestra dos campos, uno necesario llamado "required-structure" y el otro opcional llamado "optional-structure" :

```
{
  "type": "object",
  "properties": {
    "required-structure": {
      "type": "string",
      "maxLength": 8
    },
    "optional-structure": {
      "type": "string",
      "maxLength": 8
    }
  }
},
```

```
"required": [
  "required-structure"
]
```

La estructura COBOL generada muestra ambos campos, pero la segunda va precedida de "optional-structure-num" que es un recuento de enteros de los elementos, con 0 representando a ninguno y 1 que está presente. El valor se establece para indicar si "optional-structure" contiene datos válidos o no.

```
03 OutputData.
06 required-structure PIC X(8).
06 optional-structure-num PIC S9(9) COMP-5 SYNC.
06 optional-structure PIC X(8).
```

Matrices de variables de elementos

XML puede contener una matriz con distintos números de elementos. En general, los documentos WSDL y esquemas XML que contienen varios números de elementos no se correlacionan de forma eficiente en una única estructura de datos de lenguaje de alto nivel. CICS utiliza correlaciones basadas en contenedores o correlaciones en línea para gestionar varios elementos en XML.

Una matriz con varios elementos se representa en el esquema XML utilizando los atributos minOccurs y maxOccurs en la declaración del elemento:

- El atributo minOccurs especifica el número de veces mínimo que se puede dar el elemento. Puede tener un valor de 0 o un entero positivo.
- El atributo maxOccurs especifica el número de veces máximo que se puede dar el elemento. Puede tener un valor de un entero positivo mayor o igual al valor del atributo minOccurs. También puede tomar un valor de unbounded , que indica que no se aplica límite superior al número de veces que se puede dar el elemento.
- El valor predeterminado para ambos atributos es 1.

Este ejemplo indica una serie de 8 bytes que es opcional; es decir, que puede que no ocurra nunca o que ocurra una vez en el mensaje SOAP o XML de aplicación:

```
<xsd:element name="component"
  minOccurs="0" maxOccurs="1">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:length value="8"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

El ejemplo siguiente indica una serie de 8 bytes que debe tener lugar al menos una vez:

```
<xsd:element name="component"
  minOccurs="1" maxOccurs="unbounded">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:length value="8"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

En general, los documentos WSDL que contienen varios números de elementos no se correlacionan de forma eficiente en una única estructura de datos de lenguaje de alto nivel. Por lo tanto, para gestionar estos casos, CICS utiliza una serie de

estructuras de datos conectadas que se pasan al programa de aplicación en una serie de contenedores. Estas estructuras se utilizan como entrada y salida de la aplicación:

- Cuando CICS transforma XML en datos de aplicación, llena estas estructuras con los datos de aplicación y la aplicación los lee.
- Cuando CICS transforma los datos de aplicación en XML, lee los datos de aplicación en las estructuras que ha estado llenando la aplicación.

El formato de estas estructuras de datos se explica mejor con una serie de ejemplos. El XML puede ser de un mensaje SOAP o de una aplicación. Estos ejemplos utilizan una matriz de campos simples de 8 bytes. Sin embargo, el modelo soporta matrices de tipos de datos complejos y matrices de tipos de datos que contienen otras matrices.

Número fijo de elementos

El primer ejemplo ilustra un elemento que se produce tres veces exactamente:

```
<xsd:element name="component"
minOccurs="3" maxOccurs="3">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:length value="8"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

En este ejemplo, debido a que el número de veces que se da un elemento se conoce de antemano, se puede representar como una matriz de longitud fija en una sencilla declaración COBOL (o el equivalente en otros lenguajes):

```
05 component PIC X(8) OCCURS 3 TIMES
```

Número variado de elementos en el nivel de correlación 2 e inferior

Este ejemplo ilustra un elemento obligatorio que puede encontrarse de una a cinco veces:

```
<xsd:element name="component"
minOccurs="1" maxOccurs="5">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:length value="8"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

La estructura de datos principal contiene una declaración de dos campos. Cuando CICS transforma el XML en datos binarios, el primer campo `component-num` contiene el número de veces que aparece el elemento en el XML, y el segundo campo, `component-cont`, contiene el nombre del contenedor:

```
05 component-num PIC S9(9) COMP-5
05 component-cont PIC X(16)
```

Una segunda estructura de datos contiene la declaración del elemento:

```
01 DFHWS-component
02 component PIC X(8)
```

Debe examinar el valor de `component-num` (que contendrá un valor en el rango de 1 a 5) para averiguar cuántas veces se da el elemento. El contenido del elemento está

en el contenedor denominado `component-cont`; el contenedor tiene una matriz de elementos, donde cada elemento está correlacionado por la estructura de datos `DFHWS-component`.

Si `minOccurs="0"` y `maxOccurs="1"`, el elemento es opcional. Para procesar la estructura de datos en el programa de aplicación, debe examinar el valor de `component-num`:

- Si es cero, el mensaje no tiene elemento de componente y el contenido de `component-cont` no está definido.
- Si es uno, el elemento de componente está en el contenedor denominado `component-cont`.

El contenido del contenedor está correlacionado por la estructura de datos `DFHWS-component`.

Nota: Si el mensaje SOAP consta de un único elemento recurrente, `DFHWS2LS` genera dos estructuras de lenguaje. La estructura de lenguaje principal contiene el número de elementos en la matriz y el nombre del contenedor que tiene la matriz de elementos. La segunda estructura de lenguaje correlaciona una única instancia de elemento recurrente.

Número variado de elementos en el nivel de correlación 2.1 y superior

En el nivel de correlación 2.1 y superior, puede utilizar el parámetro **INLINE-MAXOCCURS-LIMIT** en los asistentes de CICS. El parámetro **INLINE-MAXOCCURS-LIMIT** especifica la forma en la que se manejan varios elementos. Las opciones de correlación para un número variado de elementos son correlaciones basadas en contenedores, descritas en “Número variado de elementos en el nivel de correlación 2 e inferior” en la página 350, o correlaciones en línea. El *valor* de este parámetro puede ser un entero positivo en el rango de 0 a 32767:

- El valor predeterminado de **INLINE-MAXOCCURS-LIMIT** es 1, que garantiza que los elementos opcionales están correlacionados en línea.
- Un valor de 0 para el parámetro **INLINE-MAXOCCURS-LIMIT** impide la correlación en línea.
- Si `maxOccurs` es menor o igual al valor de **INLINE-MAXOCCURS-LIMIT**, se utiliza la correlación en línea.
- Si `maxOccurs` es mayor que el valor de **INLINE-MAXOCCURS-LIMIT**, se utiliza la correlación basada en contenedores.

La correlación de un número variado de elementos en línea da como resultado la generación de una matriz, como sucede con el ejemplo de aparición fija anterior, y un contador. El campo `component-num` indica cuántas instancias del elemento están presentes y si están señaladas por una matriz. Para el ejemplos mostrado en “Número variado de elementos en el nivel de correlación 2 e inferior” en la página 350, cuando **INLINE-MAXOCCURS-LIMIT** es menor o igual a 5, la estructura de datos generada es como ésta:

```
05 component-num PIC S9(9) COMP-5 SYNC.  
05 component OCCURS 5 PIC X(8).
```

El primer campo, `component-num`, es idéntico a la salida para el ejemplo de correlación basada en contenedores de la sección anterior. El segundo campo contiene una matriz de longitud 5 que es lo suficientemente larga para contener el número máximo de elementos que puede generar.

La correlación en línea difiere de la correlación basada en contenedores, que almacena el número de apariciones del elemento y el nombre del contenedor donde están colocados los datos, porque almacena todos los datos en el contenedor actual. Almacenar los datos en el contenedor actual normalmente mejorará el rendimiento y hará que la correlación en línea sea preferible.

Matrices de variables anidadas

Los esquemas XML y documentos WSDL complejos pueden contener elementos recurrentes variables, que a su vez contienen elementos recurrentes variables. En este caso, la estructura descrita se amplía más allá de los dos niveles descritos en los ejemplos.

Este ejemplo ilustra en elemento opcional denominado <component2> que se anida en un elemento obligatorio denominado <component1>, donde el elemento obligatorio puede encontrarse de una a cinco veces:

```
<xsd:element name="component1"
minOccurs="1" maxOccurs="5">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="component2"
minOccurs="0" maxOccurs="1">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:length value="8"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

La estructura de datos de nivel superior es exactamente la misma que en los ejemplos anteriores:

```
05 component1-num PIC S9(9) COMP-5
05 component1-cont PIC X(16)
```

Sin embargo, la segunda estructura de datos contiene estos elementos:

```
01 DFHWS-component1
02 component2-num PIC S9(9) COMP-5
02 component2-cont PIC X(16)
```

Una estructura de tercer nivel contiene estos elementos:

```
01 DFHWS-component2
02 component2 PIC X(8)
```

El número de apariciones del elemento más externo "component1" está en component1-num.

El contenedor nombrado en component1-cont contiene una matriz con varias instancias de la segunda estructura de datos DFHWS-component1.

Cada instancia de component2-cont nombra un contenedor diferente, cada uno de los cuales contiene la estructura de datos correlacionada por la estructura de tercer nivel DFHWS-component2.

Para ilustrar esta estructura, tenga en cuenta el fragmento de XML que coincide con el ejemplo:

```

<component1><component2>
string1
</component2></component1>
<component1><component2>
string2
</component2></component1>
<component1></component1>

```

<component1> se da tres veces. Las dos primeras contienen cada una una instancia de <component2>; la tercera instancia no.

En la estructura de datos de nivel superior, component1-num contiene un valor de 3. El contenedor nombrado en component1-cont tiene tres instancias de DFHWS-component1 :

1. En la primera, component2-num tiene un valor de 1, y el contenedor nombrado en component2-cont contiene *string1*.
2. En la segunda, component2-num tiene un valor de 1, y el contenedor nombrado en component2-cont contiene *string2*.
3. En la tercera, component2-num tiene un valor de 0, y el contenido de component2-cont no está definido.

En esta instancia, la estructura de datos completa se representa por medio de cuatro contenedores en total:

- La estructura de datos raíz en el contenedor DFHWS-DATA
- El contenedor nombrado en component1-cont
- Los dos contenedores nombrados en las dos primeras instancias de component2-cont.

Estructuras opcionales y xsd:choice

DFHWS2LS y DFHSC2LS soportan el uso de maxOccurs y minOccurs en elementos <xsd:sequence>, <xsd:choice>, y <xsd:all> sólo en el nivel de correlación 2.1 y superior, donde los atributos minOccurs y maxOccurs están configurados en minOccurs="0" y maxOccurs="1".

Los asistentes generan correlaciones que tratan estos elementos como si cada elemento hijo en ellos fuera opcional. Cuando implementa una aplicación con estos elementos, asegúrese de que la aplicación no genera combinaciones no válidas de opciones. Cada uno de los elementos tiene su propio campo count en la estructura de lenguajes generada, estos campos deben establecerse todos en "0" o todos en "1". Cualquier otra combinación de valores no es válida; excepto con elementos <xsd:choice>.

Los elementos <xsd:choice> indican que sólo se puede utilizar una de las opciones del elemento. Esto se soporta en todos los niveles de correlación. Los asistentes gestionan cada una de las opciones en un <xsd:choice> aunque es un elemento <xsd:sequence> con minOccurs="0" y maxOccurs="1". Tenga cuidado cuando implementa una aplicación utilizando el elemento <xsd:choice> para asegurarse de que la aplicación no genera combinaciones no válidas de las opciones. Cada uno de los elementos tiene su propio campo count en la estructura de lenguajes generada, exactamente uno debe estar establecido en '1' y los demás establecidos en '0'. Cualquier otra combinación de valores no es válida, excepto cuando el elemento <xsd:choice> es opcional, en cuyo caso es válido para todos los campos que esté establecido en '0'.

Soporte para atributos XML

Los esquemas XML pueden especificar atributos permitidos o necesarios en XML. Los programas de utilidad del asistente de CICS, DFHWS2LS y DFHSC2LS, ignoran los atributos XML de forma predeterminada. Para procesar atributos XML que se definen en el esquema XML, el valor del parámetro **MAPPING-LEVEL** debe ser 1.1 o superior.

Atributos opcionales

Los atributos pueden ser opcionales o necesarios y se pueden asociar a cualquier elemento en el mensaje SOAP o XML para una aplicación. Para cada atributo opcional definido en el esquema, se generan dos campos en la estructura de lenguaje adecuada:

1. Un indicador de existencia; este campo se trata como un tipo de datos booleano y, normalmente, tiene 1 byte de longitud.
2. Un valor; este campo se correlaciona de la misma forma que un elemento XML de un tipo equivalentemente. Por ejemplo, un atributo de tipo NMTOKEN se correlaciona de la misma forma que un elemento XML de tipo NMTOKEN.

Los campos de valor y existencia del atributo aparecen en la estructura de lenguaje generada antes del campo para el elemento al que están asociados. Los atributos inesperado que aparecen en el documento de instancia se ignoran.

Por ejemplo, tenga en cuenta la siguiente definición de atributo de esquema:

```
<xsd:attribute name="age" type="xsd:short"
use="optional" />
```

Este atributo opcional se correlaciona con la siguiente estructura COBOL:

```
05 attr-age-exist PIC X DISPLAY
05 attr-age-value PIC S9999 COMP-5 SYNC
```

Proceso de tiempo de ejecución de atributos opcionales

El siguiente proceso de tiempo de ejecución tiene lugar para los atributos opcionales:

- Si el atributo está presente, se establece el indicador de existencia y el valor se correlaciona.
- Si el atributo no está presente, el indicador de existencia no se establece.
- Si el atributo tiene un valor predeterminado y está presente, el valor se correlaciona.
- Si el atributo tiene un valor predeterminado y no está presente, el valor predeterminado se correlaciona.

Los atributos opcionales que tienen valores predeterminados se tratan como atributos necesarios.

Cuando CICS transforma los datos en XML, tiene lugar el siguiente proceso de tiempo de ejecución:

- Si se establece el indicador de existencia, el atributo se transforma y se incluye en el XML.
- Si el distintivo de existencia no se establece, el atributo no se incluye en el XML.

Atributos necesarios y proceso de tiempo de ejecución

Para cada atributo necesario, sólo se genera el campo de valor en la estructura de lenguaje adecuada.

Si el atributo está presente en el XML, el valor se correlaciona. Si el atributo no está presente, se produce el siguiente proceso:

- Si la aplicación es un proveedor de servicios web, CICS genera un mensaje de error SOAP que indica que hay un error en el mensaje SOAP de cliente.
- Si la aplicación es un solicitante de servicio web, CICS emite un mensaje y devuelve una respuesta de error de conversión con un código RESP2 de 13 a la aplicación.
- Si la aplicación está utilizando el mandato **TRANSFORM XMLTODATA**, CICS emite un mensaje y devuelve una respuesta de solicitud no válida con un código RESP2 de 3 a la aplicación.

Cuando CICS produce un mensaje SOAP basado en el contenido de un COMMAREA o un contenedor, el atributo se transforma y se incluye en el mensaje. Cuando una aplicación utiliza el mandato **TRANSFORM DATATOXML**, CICS también transforma el atributo y lo incluye en el XML.

El atributo nillable

El atributo nillable es un atributo especial que puede aparecer en un `xsd:element` en un esquema XML. Especifica que el atributo `xsi:nil` es válido para el elemento en el XML. Si un elemento tiene especificado el atributo `xsi:nil`, indica que el elemento está presente pero no tiene valor y, por lo tanto, no tiene contenido asociado a él.

Si un esquema XML ha definido el atributo nillable como verdadero, se correlaciona como un atributo necesario que toma un valor booleano.

Cuando CICS recibe un mensaje SOAP o tiene que transformar XML para una aplicación que contiene un atributo `xsi:nil`, el valor del atributo es verdadero o falso. Si el valor es verdadero, la aplicación debe ignorar los valores del elemento o elementos anidados en el ámbito del atributo `xsi:nil`.

Cuando CICS produce un mensaje SOAP o XML basado en el contenido de un COMMAREA o de un contenedor para el que el valor para el atributo `xsi:nil` es verdadero, se produce el siguiente proceso:

- El atributo `xsi:nil` se genera en el XML o mensaje SOAP.
- El valor del elemento asociado se ignora.
- Cualquier elemento anidado dentro del elemento se ignora.

Ejemplo de mensaje SOAP

Tenga en cuenta el siguiente esquema XML de ejemplo, que puede ser parte de un documento WSDL:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="root" nillable="true">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element nillable="true" name="num" type="xsd:int" maxOccurs="3"
          minOccurs="3"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```



```

</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>

```

Aquí tiene un ejemplo de un mensaje SOAP parcial que se ajusta a este esquema:

```

<root xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <num xsi:nil="true"/>
  <num>15</num>
  <num xsi:nil="true"/>
</root>

```

En COBOL, este mensaje SOAP se correlaciona con estos elementos:

```

05 root
10 attr-nil-root-value PIC X DISPLAY
10 num OCCURS 3
15 num1 PIC S9(9) COMP-5 SYNC
15 attr-nil-num-value PIC X DISPLAY
10 filler PIC X(3)

```

Soporte para <xsd:any> y xsd:anyType

DFHWS2LS y DFHSC2LS soportan el uso de <xsd:any> y xsd:anyType en el esquema XML. Puede utilizar el elemento de esquema XML <xsd:any> para describir una sección de un documento XML con contenido no definido.

xsd:anyType es el tipo de datos base desde el que se derivan todos los tipos de datos simples y complejos; no tiene restricciones en el contenido de los datos.

Antes de poder utilizar <xsd:any> y xsd:anyType con los asistentes de CICS, establezca los siguientes parámetros:

- Establezca el parámetro **MAPPING-LEVEL** en 2.1 o superior.
- Para una aplicación de proveedor de servicios web, establezca el parámetro **PGMINT** en CHANNEL.

Ejemplo de <xsd:any>

Este ejemplo utiliza un elemento <xsd:any> para describir contenido XML no construido opcional después de la etiqueta "Surname" en la etiqueta "Customer":

```

<xsd:element name="Customer">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Title" type="xsd:string"/>
      <xsd:element name="FirstName" type="xsd:string"/>
      <xsd:element name="Surname" type="xsd:string"/>
      <xsd:any minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

```

Un mensaje SOAP de ejemplo que se ajusta a este esquema XML es:

```

<xml version='1.0' encoding='UTF-8'?>
  <SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body>
      <Customer xmlns="http://www.example.org/anyExample">
        <Title xmlns="">Mr</Title>
        <FirstName xmlns="">John</FirstName>
        <Surname xmlns="">Smith</Surname>
        <ExtraInformation xmlns="http://www.example.org/ExtraInformation">
          <!-- This 'ExtraInformation' tag is associated with the optional xsd:any
            from the XML schema.
            It can contain any well formed XML. -->

```

```

<ExampleField1>one</ExampleField1>
<ExampleField2>two</ExampleField2>
</ExtraInformation>
</Customer>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

Si este mensaje SOAP se envía a CICS , CICS llena el contenedor Customer-xml-cont con los siguientes datos XML:

```

<ExtraInformation
  xmlns="http://www.example.org/ExtraInformation">
  <!-- This 'ExtraInformation' tag is associated with the optional xsd:any
  from the XML schema.
  It can contain any well formed XML. -->
  <ExampleField1>one</ExampleField1>
  <ExampleField2>two</ExampleField2>
</ExtraInformation>

```

CICS también llena el contenedor Customer-xmlns-cont con las siguientes declaraciones de espacio de nombres XML que está en el ámbito; estas declaraciones están separadas por un espacio:

```

xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns="http://www.example.org/anyExample"

```

Ejemplo de xsd:anyType

El xsd:anyType es el tipo de datos base del que se derivan todos los tipos de datos simples y complejos. No restringe el contenido de datos. Si no especifica un tipo de datos, utiliza los valores predeterminados de xsd:anyType; por ejemplo, estos dos fragmentos XML son equivalentes:

```

<xsd:element name="Name" type="xsd:anyType"/>
<xsd:element name="Name"/>

```

Estructuras de lenguaje generadas

Las estructuras de lenguaje generadas para <xsd:any> o xsd:anyType toman el formato siguiente en COBOL y un formato equivalente para otros lenguajes:

elementName-xml-cont PIC X(16)

El nombre de un contenedor que tiene el XML sin procesar. Cuando CICS procesa un mensaje SOAP de entrada, coloca el subconjunto del mensaje SOAP que define <xsd:any> o xsd:anyType en este contenedor. La aplicación puede procesar los datos XML sólo de forma nativa. La aplicación debe generar el XML, llenar este contenedor y proporcionar el nombre de contenedor.

Este contenedor debe llenarse en modalidad de texto. Si CICS llena este contenedor, lo hace utilizando la misma variante de EBCDIC que el servicio web tiene definida para utilizar. Los caracteres que no existen en la página de códigos EBCDIC de destino se sustituyen por caracteres sustitutos; aunque el contenedor lo lea una aplicación en UTF-8.

elementName-xmlns-cont PIC X(16)

El nombre de un contenedor que tiene cualquier declaración de prefijo de espacio de nombres que esté en el ámbito. El contenido de este contenedor es similar al contenido del contenedor DFHWS-XMLNS, excepto que incluye todas las declaraciones de espacio de nombres que están en el ámbito y que son relevantes, en lugar de sólo el subconjunto de la etiqueta Sobre SOAP.

Este contenedor debe llenarse en modalidad de texto. Si CICS llena este contenedor, lo hace utilizando la misma variante de EBCDIC que el servicio web tiene definida para utilizar. Los caracteres que no existen en la página de códigos EBCDIC de destino se sustituyen por caracteres sustitutos; aunque el contenedor lo lea una aplicación en UTF-8.

Este contenedor se utiliza sólo cuando los mensajes SOAP de proceso se envían a CICS. Si la aplicación intenta proporcionar un contenedor con declaraciones de espacio de nombres cuando se genera un mensaje SOAP de salida, CICS ignora el contenedor y su contenido. CICS requiere que el XML proporcionado por la aplicación sea completamente autocontenido con respecto a las declaraciones de espacio de nombres.

El nombre del elemento XML que contiene el elemento `<xsd:any>` se incluye en los nombres de variable que se generan para el elemento `<xsd:any>`. En el ejemplo `<xsd:any>`, el elemento `<xsd:any>` se anida dentro del elemento `<xsd:element name="Customer">` y los nombres de variables que se generan para el elemento `<xsd:any>` son `Customer-xml-cont PIC X(16)` y `Customer-xmlns-cont PIC X(16)`.

Para un tipo `xsd:anyType`, se utiliza el nombre de elemento XML directo; en el ejemplo `xsd:anyType` anterior, los nombres de variable son `Name-xml-cont PIC X(16)` y `Name-xmlns-cont PIC X(16)`.

Soporte para `<xsd:choice>`

Un elemento `<xsd:choice>` indica que sólo se puede utilizar una de las opciones del elemento. Los asistentes de CICS proporcionan varios grados de soporte para elementos `<xsd:choice>` a distintos niveles de correlación.

Soporte para `<xsd:choice>` a nivel de correlación 2.2 y superior

En el nivel de correlación 2.2 y, DFHWS2LS y DFHSC2LS proporcionan un soporte mejorado para los elementos `<xsd:choice>`. Los asistentes generan un nuevo contenedor que almacena el valor asociado al elemento `<xsd:choice>`. Los asistentes generan estructuras de lenguaje que contienen un contenedor nuevo y un campo adicional:

fieldname -enum

El campo discriminante para indicar qué opciones de `<xsd:choice>` utilizará el elemento.

fieldname -cont

El nombre del contenedor que almacena la opción que se va a utilizar. Se genera una estructura de lenguaje adicional para correlacionar el valor de la opción.

El siguiente fragmento de esquema XML incluye un elemento `<xsd:choice>`:

```
<xsd:element name="choiceExample">
  <xsd:complexType>
    <xsd:choice>
      <xsd:element name="option1" type="xsd:string" />
      <xsd:element name="option2" type="xsd:int" />
      <xsd:element name="option3" type="xsd:short" maxOccurs="2" minOccurs="2" />
    </xsd:choice>
  </xsd:complexType>
</xsd:element>
```

Si este fragmento de esquema XML se procesa a un nivel de correlación 2.2 o superior, el asistente genera las siguientes estructuras de lenguaje COBOL:

```

03 choiceExample.
06 choiceExample-enum PIC X DISPLAY.
88 empty VALUE X'00'.
88 option1 VALUE X'01'.
88 option2 VALUE X'02'.
88 option3 VALUE X'03'.
06 choiceExample-cont PIC X(16).

01 Example-option1.
03 option1-length PIC S9999 COMP-5 SYNC.
03 option1 PIC X(255).

01 Example-option2.
03 option2 PIC S9(9) COMP-5 SYNC.

01 Example-option3.
03 option3 OCCURS 2 PIC S9999 COMP-5 SYNC.

```

Limitaciones para <xsd:choice> a nivel de correlación 2.2 y superior

DFHSC2LS y DFHWS2LS no soportan elemento <xsd:choice> anidados; por ejemplo, no se soporta el siguiente XML:

```

<xsd:choice>
<xsd:element name="name1" type="string"/>
<xsd:choice>
<xsd:element name="name2a" type="string"/>
<xsd:element name="name2b" type="string"/>
</xsd:choice>
</xsd:choice>

```

DFHSC2LS y DFHWS2LS no soportan elementos <xsd:choice> recurrente; por ejemplo, no se soporta el siguiente XML:

```

<xsd:choice maxOccurs="2">
<xsd:element name="name1" type="string"/>
</xsd:choice>

```

DFHSC2LS y DFHWS2LS soportan un máximo de 255 opciones en un elemento <xsd:choice>.

Soporte para <xsd:choice> a nivel de correlación 2.1 e inferior

En el nivel de correlación 2.1 e inferior, DFHWS2LS proporciona soporte limitado para elementos <xsd:choice>. DFHWS2LS trata cada una de las opciones del elemento <xsd:choice> aunque es un elemento <xsd:sequence> que se puede dar como máximo una vez.

Sólo se puede utilizar una de las opciones en un elemento <xsd:choice>, así que tenga cuidado cuando implemente una aplicación utilizando el elemento <xsd:choice> ya que genera sólo combinaciones válidas de opciones. Cada uno de los elementos tiene su propio campo count en la estructura de lenguajes generada, exactamente uno de los cuales debe establecerse en 1 y los demás deben establecerse en 0. Cualquier otra combinación de valores es incorrecta, excepto cuando <xsd:choice> es opcional, en cuyo caso, es válido para todos los campos que se van a establecer en 0.

Soporte para <xsd:sequence>

El uso de elementos <xsd:sequence> con los asistentes de CICS se soporta con varias limitaciones.

El uso de los atributos minOccurs y maxOccurs no se soporta para el elemento <xsd:sequence>. Las excepciones a esta regla son cuando minOccurs="0" y maxOccurs="1" o minOccurs="1" y maxOccurs="1".

Los asistentes de CICS no permiten el uso de dos elementos con el mismo nombre en el mismo elemento <xsd:sequence>. Por ejemplo, la secuencia {a, b, c, a} es rechazada por los asistentes de CICS. Para evitar esta limitación, sustituya la secuencia por {a maxOccurs="2", b, c}.

Los asistentes de CICS permiten un sólo elemento <xsd:any>, o un elemento que se trate como un <xsd:any>, en el mismo elemento <xsd:sequence>. Los elementos que se tratan como elementos <xsd:any> incluyen elementos abstractos que no tienen un grupo de sustitución definido.

Al analizar XML, CICS trata los elementos <xsd:sequence> como elementos <xsd:all>; esto significa que CICS no comprueba si el orden de los elementos en la secuencia es correcto. Por ejemplo, si el esquema define una secuencia de {a, b, c}, CICS soporta el XML que contiene a, b, y c en cualquier orden. Sin embargo, cuando CICS genera XML, el orden de los elementos en <xsd:sequence> se conserva siempre.

CICS no detecta automáticamente los datos XML que faltan. Por ejemplo, si se define un elemento en <xsd:sequence> como obligatorio, (minOccurs="1" maxOccurs="1"), pero no se produce en el documento XML, CICS informará de este problema sólo si está habilitada la validación en tiempo real

Soporte para los grupos de sustitución

Puede utilizar un grupo de sustitución para definir un grupo de elementos XML que son intercambiables. Los asistentes de CICS proporcionan soporte para los grupos de sustitución a nivel de correlación 2.2 y superior.

A nivel de correlación 2.2 y superior, DFHSC2LS y DFHWS2LS soportan los grupos de sustitución utilizando correlaciones similares a las utilizadas para los elementos <xsd:choice>. El asistente genera un campo de enumeración y un nombre de contenedor nuevo en la estructura de lenguaje.

El siguiente fragmento de esquema XML incluye una matriz de dos elementos subGroupParent, cada uno de los cuales puede sustituirse por replacementOption1 o replacementOption2:

```
<xsd:element name="subGroupExample"
  >
  <xsd:complexType>
  <xsd:sequence>
  <xsd:element ref="subGroupParent" maxOccurs="2" minOccurs="2" />
  </xsd:sequence>
  </xsd:complexType>
  </xsd:element>

  <xsd:element name="subGroupParent" type="xsd:anySimpleType" />
  <xsd:element name="replacementOption1" type="xsd:int"
  substitutionGroup="subGroupParent" />
  <xsd:element name="replacementOption2" type="xsd:short"
  substitutionGroup="subGroupParent" />
```

Procesando este fragmento XML con el asistente genera las siguientes estructuras de lenguaje COBOL:

```

03 subGroupExample.
06 subGroupParent OCCURS2.
09 subGroupExample-enum PIC X DISPLAY.
88 empty VALUE X '00'.
88 replacementOption1 VALUE X '01'.
88 replacementOption2 VALUE X '02'.
88 subGroupParent VALUE X '03'.
09 subGroupExample-cont PIC X (16).

```

```

01 Example-replacementOption1.
03 replacementOption1 PIC S9(9) COMP-5 SYNC.

```

```

01 Example-replacementOption2.
03 replacementOption2 PIC S9999 COMP-5 SYNC.

```

```

01 Example-subGroupParent.
03 subGroupParent-length PIC S9999 COMP-5 SYNC.
03 subGroupParent PIC X(255).

```

Para obtener más información sobre los grupos de sustitución, consulte *W3C XML Schema Part 1: Structures Second Edition specification* : http://www.w3.org/TR/xmlschema-1/#Elements_Equivalence_Class

Soporte para elementos abstractos y tipos de datos abstractos

Los asistentes de CICS proporcionan soporte para elementos abstractos y tipos de datos abstractos a nivel de correlación 2.2 y superior. Los asistentes de CICS correlacionan elementos abstractos y tipos de datos abstractos de forma similar a los grupos de sustitución.

Soporte para elementos abstractos a nivel de correlación 2.2 y superior

En el nivel de correlación 2.2 y superior, DFHSC2LS y DFHWS2LS tratan los elementos abstractos de casi la misma forma que los grupos de sustitución excepto que el elemento abstracto no es un miembro válido del grupo. Si no hay elementos sustituibles, el elemento abstracto se trata como un elemento `<xsd:any>` y utiliza las mismas correlaciones que el elemento `<xsd:any>` en el nivel de correlación 2.1.

El siguiente fragmento de esquema XML especifica dos opciones que se pueden utilizar en lugar del elemento abstracto. El elemento abstracto en sí mismo no es una opción válida:

```

<xsd:element name="abstractElementExample" >
<xsd:complexType>
<xsd:sequence>
<xsd:element ref="abstractElementParent" maxOccurs="2" minOccurs="2" />
</xsd:sequence>
</xsd:complexType>
</xsd:element>

<xsd:element name="abstractElementParent" type="xsd:anySimpleType"
abstract="true" />
<xsd:element name="replacementOption1" type="xsd:int"
substitutionGroup="abstractElementParent" />
<xsd:element name="replacementOption2" type="xsd:short"
substitutionGroup="abstractElementParent" />

```

Procesando este fragmento XML con el asistente genera las siguientes estructuras de lenguaje COBOL:

```

03 abstractElementExample.
06 abstractElementParent OCCURS 2.
09 abstractElementExample-enum PIC X DISPLAY.
88 empty VALUE X '00'.
88 replacementOption1 VALUE X '01'.
88 replacementOption2 VALUE X '02'.
09 abstractElementExample-cont PIC X (16).

```

```

01 Example-replacementOption1.
03 replacementOption1 PIC S9(9) COMP-5 SYNC.

```

```

01 Example-replacementOption2.
03 replacementOption2 PIC S9999 COMP-5 SYNC.

```

Para obtener más información sobre los elementos abstractos, consulte W3C XML Schema Part 0: Primer Second Edition specification : <http://www.w3.org/TR/xmlschema-0/#SubsGroups>

Soporte para tipos de elementos abstractos a nivel de correlación 2.2 y superior

En el nivel de correlación 2.2 y superior, DFHSC2LS y DFHWS2LS tratan los tipos de datos abstractos como grupos de sustitución. El asistente genera un campo de enumeración y un nombre de contenedor nuevo en la estructura de lenguaje.

El siguiente fragmento de esquema XML especifica dos alternativas que se pueden utilizar en lugar del tipo abstracto:

```

<xsd:element name="AbstractDataTypeExample"
type="abstractDataType" />

<xsd:complexType name="abstractDataType" abstract="true">
<xsd:simpleContent>
<xsd:extension base="xsd:string" />
</xsd:simpleContent>
</xsd:complexType>
<xsd:complexType name="option1">
<xsd:simpleContent>
<xsd:restriction base="abstractDataType">
<xsd:length value="5" />
</xsd:restriction>
</xsd:simpleContent>
</xsd:complexType>
<xsd:complexType name="option2">
<xsd:simpleContent>
<xsd:restriction base="abstractDataType">
<xsd:length value="10" />
</xsd:restriction>
</xsd:simpleContent>
</xsd:complexType>

```

Procesando este fragmento XML con el asistente genera las siguientes estructuras de lenguaje COBOL:

```

03 AbstractDataTypeExamp-enum PIC X DISPLAY.
88 empty VALUE X'00'.
88 option1 VALUE X'01'.
88 option2 VALUE X'02'.
03 AbstractDataTypeExamp-cont PIC X(16).

```

Las estructuras de lenguaje se generan en libros de copias individuales. La estructura de lenguaje generada para option1 se genera en un libro de copias:

```

03 option1 PIC X(5).

```

La estructura de lenguaje para option2 se genera en un libro de copias diferente:
03 option2 PIC X(10).

Para obtener más información sobre los tipos de datos abstractos, consulte W3C *XML Schema Part 0: Primer Second Edition specification* : <http://www.w3.org/TR/xmlschema-0/#SubsGroups>

Cómo gestionar el contenido que se repite variablemente en COBOL

En COBOL, no puede procesar el contenido que se repite variablemente utilizando aritmética de punteo para dirigir cada instancia de datos. Otros lenguajes de programación no tienen esta limitación. Este ejemplo le muestra cómo gestionar el contenido que se repite variablemente en COBOL para una aplicación de servicio web.

Esta técnica también se aplica para transformar XML en datos de aplicación utilizando los mandatos de API **TRANSFORM**. El siguiente documento WSDL de ejemplo representa un servicio web con datos de aplicación que constan de una serie de 8 caracteres que recurre un número de veces variable:

```
<?xml version="1.0"?>
  <definitions name="ExampleWSDL"
    targetNamespace="http://www.example.org/variablyRepeatingData/"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:tns="http://www.example.org/variablyRepeatingData/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">

    <types>
      <xsd:schema targetNamespace="http://www.example.org/variablyRepeatingData/">
        <xsd:element name="applicationData">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="component" minOccurs="1" maxOccurs="unbounded">
                <xsd:simpleType>
                  <xsd:restriction base="xsd:string">
                    <xsd:length value="8"/>
                  </xsd:restriction>
                </xsd:simpleType>
              </xsd:element>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:schema>
    </types>

    <message name="exampleMessage">
      <part element="tns:applicationData" name="messagePart"/>
    </message>

    <portType name="examplePortType">
      <operation name="exampleOperation">
        <input message="tns:exampleMessage"/>
        <output message="tns:exampleMessage"/>
      </operation>
    </portType>

    <binding name="exampleBinding" type="tns:examplePortType">
      <soap:binding style="document"
        transport="http://schemas.xmlsoap.org/soap/http"/>
      <operation name="exampleOperation">
        <soap:operation soapAction="">
          <input><soap:body parts="messagePart" encodingStyle="" use="literal"/></input>
          <output><soap:body parts="messagePart" encodingStyle=""
```



```

use="literal"/></output>
</operation>
</binding>
</definitions>

```

El proceso de este documento WSDL a través de DFHWS2LS genera las siguientes estructuras de lenguaje COBOL:

```
03 applicationData.
```

```
06 component-num PIC S9(9) COMP-5 SYNC.
06 component-cont PIC X(16).
```

```
01 DFHWS-component.
03 component PIC X(8).
```

Tenga en cuenta que el campo component de 8 caracteres se define en una estructura individual llamada DFHWS-component. La estructura de datos principal se denomina applicationData y contiene dos campos, component-num y component-cont. El campo component-num indica cuántas instancias de datos component están presentes y el campo component-cont indica el nombre de un contenedor que contiene la lista concatenada de campos component.

El siguiente código COBOL muestra una forma de procesar la lista de datos recurrentes de forma variable. Utiliza una matriz de Linkage Section para direccionar instancias posteriores de datos, cada uno de los cuales se visualiza utilizando la sentencia DISPLAY:

```
IDENTIFICATION DIVISION.
PROGRAM-ID. EXVARY.
```

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
```

```
* variables de almacenamiento de trabajo
01 APP-DATA-PTR USAGE IS POINTER.
01 APP-DATA-LENGTH PIC S9(8) COMP.
01 COMPONENT-PTR USAGE IS POINTER.
01 COMPONENT-DATA-LENGTH PIC S9(8) COMP.
01 COMPONENT-COUNT PIC S9(8) COMP-4 VALUE 0.
01 COMPONENT-LENGTH PIC S9(8) COMP.
```

```
LINKAGE SECTION.
```

```
* una matriz de linkage section grande
01 BIG-ARRAY PIC X(659999).
```

```
* estructuras de datos de aplicación creadas por DFHWS2LS
* normalmente se hace referencia a esto con una sentencia COPY
01 DFHWS2LS-data.
03 applicationData.
06 component-num PIC S9(9) COMP-5 SYNC.
06 component-cont PIC X(16).
```

```
01 DFHWS-component.
03 component PIC X(8).
```

```
PROCEDURE DIVISION USING DFHEIBLK.
A-CONTROL SECTION.
A010-CONTROL.
```

```
* Obtener contenedor DFHWS-DATA
```

```

EXEC CICS GET CONTAINER('DFHWS-DATA')
SET(APP-DATA-PTR)
FLENGTH(APP-DATA-LENGTH)
END-EXEC
SET ADDRESS OF DFHWS2LS-data TO APP-DATA-PTR

* Obtener datos de componente recurrentes
EXEC CICS GET CONTAINER(component-cont)
SET(COMPONENT-PTR)
FLENGTH(COMPONENT-DATA-LENGTH)
END-EXEC

* Señalar la estructura de componente en la primera instancia de
los datos
SET ADDRESS OF DFHWS-component TO COMPONENT-PTR

* Almacenar la longitud de un único componente
MOVE LENGTH OF DFHWS-component TO COMPONENT-LENGTH

* procesar cada instancia de datos de componente de una en una
PERFORM WITH TEST AFTER
UNTIL COMPONENT-COUNT = component-num

* visualizar la instancia actual de los datos
DISPLAY 'component value is: ' component

* direccionar la siguiente instancia de datos de componente
SET ADDRESS OF BIG-ARRAY TO ADDRESS OF DFHWS-component
SET ADDRESS OF DFHWS-component
TO ADDRESS OF BIG-ARRAY (COMPONENT-LENGTH + 1:1)
ADD 1 TO COMPONENT-COUNT

* finalizar el bucle
END-PERFORM.

* Señalar la estructura de componente de vuelta a la primera instancia de
* los datos, para cualquier proceso posterior que deseemos llevar a cabo
SET ADDRESS OF DFHWS-component TO COMPONENT-PTR

* volver a CICS.

EXEC CICS
RETURN
END-EXEC

GOBACK.

```

El código anterior proporciona una solución genérica para manejar contenido que se repite de forma variable. La matriz, BIG-ARRAY, se mueve al inicio de cada componente a su vez y no permanece fijo al inicio de los datos. La estructura de datos de componente se mueve para señalar el primer byte del siguiente componente. COMPONENT-PTR puede utilizarse para recuperar la posición de inicio de los datos de componente si es necesario.

Aquí tiene un mensaje SOAP de ejemplo que se ajusta al documento WSDL:

```

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body>
<applicationData xmlns="http://www.example.org/variablyRepeatingData/">
<component xmlns="">VALUE1</component>
<component xmlns="">VALUE2</component>

```

```
<component xmlns="">VALUE3</component>
</applicationData>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Aquí tiene la salida producida por el programa COBOL cuando procesa el mensaje SOAP:

```
CPIH 20080115103151 component value is: VALUE1
CPIH 20080115103151 component value is: VALUE2
CPIH 20080115103151 component value is: VALUE3
```

Soporte para valores de longitud variable y espacio en blanco

Puede personalizar la forma en que se gestionan los valores de longitud variable y el espacio en blanco utilizando valores en los asistentes de CICS y añadiendo facetas directamente en el esquema XML.

Normalmente, el asistente XML de CICS y el asistente de servicios web de CICS correlacionan series de datos con matrices de caracteres de longitud fija; estas matrices requieren relleno con espacios o nulos. La correlación de valores de longitud variables a matrices de datos de longitud fija puede ser ineficaz y un almacenamiento de residuos. Si la longitud de datos es variable, se recomienda personalizar la manera en la que se manejan estas correlaciones.

Si está convirtiendo desde una estructura de lenguaje a un esquema XML o documento WSDL, se le recomienda especificar las facetas `whiteSpace` y `maxLength` en el esquema XML y configurar el parámetro **CHAR-VARYING-LIMIT** en los asistentes.

Si está convirtiendo desde un esquema XML o documento WSDL a una estructura de lenguaje, se recomienda establecer un valor adecuado para el parámetro **CHAR-VARYING** en los asistentes.

Nota: Los caracteres nulos ('x00') no son válidos en documentos XML. Los caracteres nulo de los datos de aplicación analizados por CICS se consideran para indicar el final de la serie y el valor se trunca. Cuando CICS genera datos de aplicación, lo hace según el valor del parámetro **CHAR-VARYING**. Por ejemplo, si se especifica la opción **CHAR-VARYING=NULL**, las series de longitud variable generadas por CICS terminan con un carácter nulo.

Correlación de valores de longitud variable desde XML a estructuras de lenguaje

Utilice facetas en el esquema XML o especifique determinados parámetros en los asistentes de CICS para personalizar la forma en la que se manejan las correlaciones entre el esquema XML o el documento WSDL y la estructura de lenguaje.

Los tipos de datos XML se pueden restringir utilizando facetas. Utilice las facetas de longitud (`length` , `maxLength` y `minLength`) y la faceta `whiteSpace` para personalizar cómo se manejan los datos de longitud variables en el XML.

length

Se utiliza para especificar si los datos son de longitud fija.

maxLength

Se utiliza para especificar la longitud máxima para el tipo de datos. Si no se establece este valor para un tipo de datos basado en series, la longitud máxima no está limitada.

minLength

Se utiliza para especificar la longitud mínima para el tipo de datos. Si no se establece este valor para un tipo de datos basado en series, la longitud mínima es 0.

whiteSpace

Se utiliza para especificar cómo se maneja el espacio en blanco alrededor de un valor de datos. Los espacios en blanco incluyen espacios, pestañas y líneas nuevas. La faceta `whiteSpace` se puede establecer en `preserve`, `replace` o `collapse`:

- Un valor de `preserve` mantiene cualquier espacio en blanco en los valores de datos.
- Un valor de `replace` significa que cualquier pestaña o línea nueva se sustituye con el número de espacios adecuados.
- Un valor de `collapse` significa que los en blanco iniciales, finales e incluidos se eliminan, y todas las pestañas, líneas nuevas y espacios consecutivos se sustituyen por caracteres de un sólo espacio.

Si la faceta `whiteSpace` no está establecida, se conserva el espacio en blanco.

Para obtener más información sobre las facetas de esquema XML, consulte el esquema de recomendación W3C *XML Schema Part 2: Datatypes Second Edition* <http://www.w3.org/TR/xmlschema-2/#facets>

Se pueden utilizar los parámetros siguientes en los asistentes de CICS, DFHSC2LS y DFHWS2LS, para alterar la forma en la que los datos de longitud variable se correlacionan desde el esquema XML a la estructura de lenguaje. Estos parámetros están disponibles en el nivel de correlación 1.2 o superior.

DEFAULT-CHAR-MAXLENGTH

Especifica la longitud de matriz predeterminada de los datos de caracteres en caracteres para correlaciones donde no está implicada la longitud en el esquema XML o el documento WSDL. El valor de este parámetro puede ser un entero positivo en el rango de 1 a 2 147 483 647.

Sin embargo, se recomienda especificar la longitud de caracteres máxima que desea que utilicen DFHSC2LS o DFHWS2LS directamente en el esquema XML o documento WSDL con la faceta `maxLength`. Especificando la longitud máxima directamente en el esquema XML o documento WSDL evita problemas asociados con tener un valor predeterminado global aplicado a todos los tipos de datos basados en series.

CHAR-VARYING-LIMIT

Especifica el tamaño máximo de los datos de caracteres de longitud variable que se correlacionan con la estructura de lenguaje. Si los datos de caracteres son mayores que el valor especificado en este parámetro, se correlacionan con un contenedor y el nombre de contenedor se utiliza en la estructura de lenguaje generada. El valor puede estar en el rango de 0 al valor predeterminado 32 767 bytes.

CHAR-VARYING

Especifica cómo se correlacionan los datos de caracteres de longitud variable. Si no especifica este parámetro, la correlación predeterminada depende del lenguaje especificado. Puede seleccionar estas opciones:

- **CHAR-VARYING=NO** especifica que los datos de caracteres de longitud variable se correlacionan como series de longitud fija.

- **CHAR-VARYING=NULL** especifica que los datos de caracteres de longitud variable se correlacionan con series terminadas en nulo.
- **CHAR-VARYING=YES** especifica que los datos de caracteres de longitud variable se correlacionan con un tipo de datos CHAR VARYING en PL/I. En lenguajes COBOL, C y C++, los datos de caracteres de longitud variable se correlacionan con una representación equivalente que consta de dos elementos relacionados: la longitud de datos y los datos.

El valor **CHAR-VARYING=YES** normalmente da como resultado el mejor rendimiento.

Correlación de valores de longitud variable desde estructuras de lenguaje a XML

Puede personalizar la forma en la que se manejan las correlaciones entre la estructura de lenguaje y el esquema XML o documento WSDL. Establezca el parámetro **CHAR-VARYING** en DFHLS2SC o DFHLS2WS, en COLLAPSE o NULL para cambiar la manera en la que se generan las matrices de caracteres.

Establezca la opción **CHAR-VARYING=NULL** que le dice a CICS que añada un carácter nulo al final de cada matriz de caracteres cuando genera XML.

Establezca la opción **CHAR-VARYING=COLLAPSE** que le dice a CICS que elimine automáticamente cualquier espacio final del final de las matrices de caracteres al generar XML. Esta opción está disponible sólo en los niveles de correlación 2.1 o superior y **CHAR-VARYING=COLLAPSE** es el valor predeterminado en el nivel de correlación 2.1 o superior para todos los lenguajes distintos a C y C++. Cuando se analiza el XML, se eliminan todos los espacios en blanco de inicio, final e incluidos.

Para obtener más información, consulte Soporte para valores de longitud variable y espacios en blanco en servicios web de CICS (nota técnica) .

Soporte para UTF-16 en datos de aplicación

Los servicios web de CICS soportan la conversión de datos de aplicación codificados con UTF-16 en XML o JSON y también de XML o JSON en datos de aplicación codificados en UTF-16. Utilice UTF-16 cuando necesite almacenar y procesar datos en varios lenguajes.

Los servicios web JSON y SOAP de CICS soportan la conversión de datos de aplicación codificados con UTF-16 en XML o JSON y también de XML o JSON en datos de aplicación codificados en UTF-16. Unicode es un esquema de codificación de anchura variable que permite a los sistemas manejar datos de forma eficaz.

UTF-16 es una codificación de ancho variable para Unicode, donde cada carácter está representado por 2 o 4 bytes. Los servicios web de CICS soportan CCSID 1200 para datos de aplicación, que es UTF-16 BE (big endian) con IBM Private Use Area. Este comportamiento es coherente con el soporte UTF-16 en todos los lenguajes soportados.

UTF-16 se soporta del nivel de correlación 4.0 en adelante. Puede personalizar cómo se convierten los datos de aplicación utilizando valores de correlación en los asistentes. Para obtener más información sobre niveles de correlación XML, consulte Mapping levels for the CICS assistants. Para obtener más información sobre niveles de correlación JSON, consulte Mapping levels for the CICS JSON assistants.

Nota: UTF-16 requiere más tiempo de proceso y menos eficiencia de almacenamiento que las codificaciones EBCDIC. Además, la mezcla de tipos de codificación provoca un proceso de tiempo de ejecución adicional.

Correlación UTF-16 de esquema XML o JSON a estructuras de lenguaje

El soporte para UTF-16 depende de cómo cree el servicio web. La correlación del esquema XML o JSON a estructuras de lenguaje, también conocida como correlación descendente, tiene las siguientes características. Si UTF-16 está habilitado, todos los campos de texto se correlacionan con campos UTF-16, mientras que los tipos de datos de visualización numérica en COBOL se correlacionan como EBCDIC. Para utilizar UTF-16, establezca el parámetro CCSID de DFHJS2LS, DFHSC2LS o DFHWS2LS en 1200.

Por ejemplo, si el siguiente fragmento de esquema XML estaba presente en WSDL:

```
<xsd:element name="myString" nillable="false">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:maxLength value="20"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

El asistente DFHWS2LS genera el campo siguiente en una estructura de lenguaje COBOL:

```
myString PIC N(
20
) USAGE NATIONAL
```

El parámetro CHAR-MULTIPLIER de los asistentes de servicios web se puede utilizar para especificar la longitud de un campo que generan los asistentes.

CHAR-MULTIPLIER

Cuando utiliza UTF-16, los únicos valores válidos para el parámetro **CHAR-MULTIPLIER** son 2 o 4 , donde 2 es el valor predeterminado.

CHAR-MULTIPLIER = 2 , donde el esquema describe una serie de maxlength x , genera PIC N(x) . El valor **CHAR-MULTIPLIER** = 2 no excluye el uso de pares de sustitución en una serie UTF-16, pero tiene repercusión en el número de caracteres que caben en el campo.

CHAR-MULTIPLIER = 4 genera PIC N($2x$) . Si **CHAR-MULTIPLIER** = 4 , el valor en tiempo de ejecución se llena si la serie incluye caracteres que se pueden expresar en una única unidad de codificación.

Correlación UTF-16 de estructuras de lenguaje a esquema XML o JSON

Correlación de una estructura de lenguaje a un esquema XML o JSON, también conocida como correlación ascendente, se gestiona de forma distinta a la correlación descendente. Si se declara una serie UTF-16 en la estructura de lenguaje, CICS interpreta los datos como datos codificados en UTF-16, de lo contrario, se presupone que los datos están en codificación EBCDIC. El parámetro CCSID para DFHLS2JS, DFHLS2SC o DFHLS2WS indica la codificación de cualquier texto EBCDIC dentro de los datos de aplicación; no debe configurarse para indicar UTF-16.

Los tipos de datos que se interpretan como caracteres UTF-16 son los siguientes: PIC N (*n*) en COBOL, WIDECHAR(*n*) en PL/I y char16_t[*n*] en C y C++.

El parámetro CHAR-USAGE de los asistentes de servicios web se puede utilizar para especificar tipos de datos.

CHAR-USAGE

En COBOL, el tipo de datos nacional, PIC N , se puede utilizar para datos UTF-16 o DBCS. Este valor lo controla la opción del compilador NSYMBOL. Debe establecer el parámetro **CHAR-USAGE** en el asistente en el mismo valor que la opción de compilador NSYMBOL para asegurarse de que los datos se manejan adecuadamente. Se establece normalmente en CHAR-USAGE=NATIONAL cuando utiliza UTF-16.

Si desea mezclar tipos de datos nacionales que contienen datos UTF-16 y DBCS en el mismo libro de copias, puede utilizar los calificadores USAGE NATIONAL o USAGE DISPLAY-1 en campos individuales.

Nota: DFHLS2WS, DFHLS2SC y DFHLS2JS no soportan la cláusula de COBOL GROUP USAGE NATIONAL.

Consulta XML desde una aplicación

Puede escribir un programa de aplicación para consultar un fragmento de XML antes de transformarlo en datos de aplicación.

Acerca de esta tarea

Si la aplicación va a procesar muchos tipos distintos de XML, es posible que desee consultar el XML para determinar qué recurso XMLTRANSFORM utilizar para transformarlo en datos de aplicación. Este mandato también puede ser útil si su XML contiene elementos <xsd:any>.

Procedimiento

1. En el programa de aplicación, utilice el mandato de API **TRANSFORM XMLTODATA** para consultar el XML:

```
EXEC CICS TRANSFORM XMLTODATA  
CHANNEL('MyChannelName')  
XMLCONTAINER('SourceContainerName')  
ELEMNAME(elementName) ELEMNAMELEN(elementNameLength)
```

Debe especificar el nombre del canal y el nombre del contenedor que contiene el XML. No tiene que especificar un recurso XMLTRANSFORM para consultar el XML. El ejemplo anterior consulta el nombre del primer elemento XML y la longitud del elemento XML. También puede consultar el tipo del primer elemento XML, la longitud del tipo y el espacio de nombres del tipo.

2. Opcional: Si la aplicación requiere el espacio de nombres del elemento XML, proporcione un área de datos donde CICS pueda escribir el valor ELEMNS.
3. Opcional: Después de consultar el XML, puede escribir lógica de aplicación para determinar qué recurso XMLTRANSFORM utilizar para transformar el XML en datos de aplicación.
4. Instale el programa de aplicación en CICS.

Resultados

CICS lee el contenedor especificado y devuelve la información sobre el elemento XML al programa de aplicación.

Manejo de XML por tipo de datos

Si el esquema XML contiene tipos de datos globales, uno o más a los que se hace referencia en XML, puede generar metadatos para soportar estos tipos de datos globales y, a continuación, analizar o transformar el XML en un programa de aplicación.

Antes de empezar

Para generar los metadatos correctos, debe ejecutar DFHSC2LS especificando el parámetro TYPES=ALL.

Acerca de esta tarea

Procedimiento

1. Para transformar datos de aplicación en XML por tipo de datos, utilice el mandato **TRANSFORM DATATOXML**:

```
EXEC CICS TRANSFORM DATATOXML
XMLTRANSFORM('
  MyXmlTransformName
')
CHANNEL('
  MyChannelName
')
DATCONTAINER('
  SourceContainerName
')
XMLCONTAINER('
  TargetContainerName
')
ELEMNAME(
  elementName
) ELEMNAMELEN(
  elementNameLength
)
ELEMNS(
  elementNamespace
) ELEMNSLEN(
  elementNamespaceLength
)
TYPENAME(
  typeName
) TYPENAMELEN(
  typeNameLen
)
TYPENS(
  typeNamespace
) TYPENSLEN(
  typeNamespaceLen
)
```

MyXmlTransformName es el nombre de 32 caracteres del recurso XMLTRANSFORM que especifica el enlace y esquema XML; *MyChannelName* es el nombre de 16 caracteres del canal que tiene los contenedores de entrada y salida; *SourceContainerName* es el nombre de 16 caracteres del contenedor de entrada que contiene los datos de aplicación y *TargetContainerName* es el nombre de 16 caracteres del contenedor de salida que CICS llena con XML. También debe especificar el nombre del elemento XML, el espacio de nombres, el tipo de datos y el espacio de nombres del tipo de datos para la transformación.

2. Para analizar el XML por tipo de datos, utilice el mandato **TRANSFORM XMLTODATA**. Las opciones que especifica en el mandato dependen de si el XML tiene un atributo `xsi:type`.

- Si el XML utiliza el atributo `xsi:type`, especifique el mandato siguiente en el programa de aplicación:

```
EXEC CICS TRANSFORM XMLTODATA
XMLTRANSFORM('
MyXmlTransformName
')
CHANNEL('
MyChannelName
')
XMLCONTAINER('
SourceContainerName
')
DATCONTAINER('
TargetContainerName
')
ELEMNAME(elementName) ELEMNAMELEN(elementNameLength)
TYPENAME(typeName) TYPENAMELEN(typeNameLength)
```

Si la aplicación también requiere un espacio de nombres del tipo de datos, añada la opción `TYPENS` al mandato de API. CICS devuelve el tipo de datos desde el atributo `xsi:type` en `TYPENAME`.

- Si el XML no utiliza el atributo `xsi:type`, el programa de aplicación puede especificar el nombre local y el espacio de nombres del tipo de datos global. Utilice el mandato siguiente en el programa de aplicación:

```
EXEC CICS TRANSFORM XMLTODATA
XMLTRANSFORM('
MyXmlTransformName
')
CHANNEL('
MyChannelName
')
XMLCONTAINER('
SourceContainerName
')
DATCONTAINER('
TargetContainerName
')
TYPENAME(
typeName
) TYPENAMELEN(
typeNameLength
)
TYPENS(
TypeNamespace
) TYPENSLEN(
typeNamespaceLen
)
```

El programa de aplicación deben especificar el nombre de tipo y el espacio de nombres pero no el nombre y el espacio de nombres. Este mandato indica que la aplicación le dice a CICS qué tipo de datos utilizar. Cuando CICS está generando XML, CICS añade siempre el atributo `xsi:type`.

3. Instale el programa de aplicación en CICS.

Qué hacer a continuación

Pruebe que el programa de aplicación genera y analiza el XML como se esperaba.

Manejo de los tipos de datos <xsd:any>

Si trabaja con un esquema XML que contiene uno o más tipos de datos <xsd:any>, los asistentes XML pueden correlacionar el tipo de datos a un par de contenedores CICS. Puede escribir un programa de aplicación para analizar el XML de los contenedores.

Antes de empezar

Debe correlacionar el esquema XML utilizando DFHSC2LS o DFHWS2LS mediante un nivel de correlación de 2.1 o superior.

Acerca de esta tarea

Cuando CICS transforma los datos en XML, coloca el XML asociado al tipo de datos <xsd:any> en el primer contenedor y las declaraciones de prefijo de espacio de nombres que están en el ámbito en el segundo contenedor.

Procedimiento

1. Para analizar los datos XML, utilice el mandato **TRANSFORM XMLTODATA** en su programa de aplicación:

```
EXEC CICS TRANSFORM XMLTODATA
XMLTRANSFORM('
  MyXmlTransformName
')
CHANNEL('
  MyChannelName
')
XMLCONTAINER('
  SourceContainerName
')
DATCONTAINER('
  TargetContainerName
')
NSCONTAINER('
  NamespacesContainerName
')
ELEMNAME(elementName) ELEMNAMELEN(elementNameLength)
```

MyXmlTransformName es el nombre de 32 caracteres del recurso XMLTRANSFORM que especifica el enlace y esquema XML; *MyChannelName* es el nombre de 16 caracteres del canal que tiene los contenedores de entrada y salida; *SourceContainerName* es el nombre de 16 caracteres del contenedor de entrada que contiene el XML y *TargetContainerName* es el nombre de 16 caracteres del contenedor de salida que llena CICS con datos de aplicación; *NamespacesContainerName* es el nombre de 16 caracteres del contenedor que llena CICS con las declaraciones de prefijo del espacio de nombres. Proporcione valores iniciales para las opciones ELEMNAME y ELEMNAMELEN. CICS devuelve el elemento XML y su longitud en las opciones ELEMNAME y ELEMNAMELEN.

2. Instale el programa de aplicación en CICS.

Qué hacer a continuación

Pruebe que el programa de aplicación analiza el XML correctamente.

Validación de transformaciones de XML

Si utiliza el asistente XML de CICS para correlacionar datos de aplicación a XML, puede especificar que las transformaciones que tienen lugar durante el tiempo de

ejecución se validan para garantizar que se ajustan al esquema que contiene el enlace XML. Puede realizar la validación en la transformación de XML a datos binarios o de datos binarios a XML.

Antes de empezar

Durante el desarrollo y prueba de la aplicación CICS, la validación completa ayuda en la detección de problemas en XML. Sin embargo, la validación completa de XML tiene una sobrecarga considerable y, por lo tanto, es desaconsejable validar XML en una aplicación de producción completamente probada.

CICS utiliza un programa Java para validar el XML en el esquema. Por lo tanto, debe tener el soporte Java habilitado en la región de CICS para llevar a cabo la validación.

Procedimiento

1. Configure el servidor JVM en la región de CICS. La clase de validador XML se puede ejecutar en un marco de trabajo OSGi o Axis2, pero no en un perfil de Liberty. CICS proporciona ejemplos para configurar rápidamente un servidor JVM que utiliza un marco de trabajo OSGi.
 - a. Instale el servidor JVM de ejemplo DFHJVMS en el grupo DFH\$OSGI o cree su propio servidor JVM. Para obtener más información, consulte el apartado *Setting up a JVM server*.
 - b. Si ha creado su propio servidor JVM, modifique la definición de programa DFHPIVAL en el grupo DFHPIVAL para hacer referencia al nombre del recurso JVMSERVER. La definición DFHPIVAL no está bloqueada y se puede editar. De forma predeterminada, la definición hace referencia a DFHJVMS.
2. Asegúrese de que el enlace XML y el esquema están en la misma ubicación en z/OS UNIX. El recurso XMLTRANSFORM define estos archivos para CICS. Puede utilizar el mandato **INQUIRE XMLTRANSFORM** para comprobar la ubicación de cada archivo.
3. Active la validación para la aplicación. En CICS Explorer, abra el recurso XMLTRANSFORM y edite el campo Estado de validación en la lista de atributos. De lo contrario, puede utilizar CEMT o SPI.

Resultados

Compruebe el registro del sistema para averiguar si la transformación de XML es válida. El mensaje DFHML0508 indica que el XML se ha validado correctamente y el mensaje DFHML0507 indica que la validación ha fallado.

Qué hacer a continuación

Cuando ya no necesite la validación XML para la aplicación, actualice el recurso XMLTRANSFORM para desactivarla.

Generación de correlaciones desde estructuras de lenguaje

Para crear XML a partir de datos de aplicación o viceversa, cree las correlaciones para describir cómo CICS transformará los datos y XML en tiempo de ejecución. Puede comenzar desde cualquier registro de datos de aplicación; por ejemplo, puede comenzar con un COMMAREA, archivo VSAM, cola de almacenamiento temporal o un registro DB2.

Antes de empezar

Antes de crear las correlaciones, debe asegurarse que se han completado estas condiciones previas:

- Debe tener una estructura de lenguaje que describa el registro de aplicación en un conjunto de datos particionados. La estructura de lenguaje se puede escribir en cualquier lenguaje de alto nivel que soporte el asistente de XML de CICS: COBOL, PL/I, C y C++. Si esta correlación es para utilizarla con un feed Atom, y está utilizando cualquiera de los campos del registro de aplicación para proporcionar metadatos para las entradas Atom (como el nombre de un autor), asegúrese de que esos campos no están anidados en la estructura de lenguaje. Puede proporcionar estructuras de campos anidados dentro de un campo que proporciona el contenido para una entrada Atom.
- Debe configurar el ID de usuario bajo en el que se ejecuta DFHLS2SC para utiliza z/OS UNIX.
- El ID de usuario debe tener permiso de lectura para acceder a la estructura de lenguaje y permiso de escritura para poner la salida en los directorios adecuados en z/OS UNIX.
- Debe asignar almacenamiento suficiente para el ID de usuario para que el ID ejecute Java. Puede utilizar cualquier versión soportada de Java. Se forma predeterminada, DFHLS2SC utiliza la versión de Java especificada en el parámetro **JAVADIR**.

Acerca de esta tarea

Utilice el asistente XML de CICS para crear las correlaciones de datos para el registro de aplicación. Para cada lenguaje de alto nivel soportado por el asistente XML de CICS, se restringen unos cuantos tipos de datos o no se soportan. El asistente XML de CICS emite mensajes de error sobre cualquier elemento no soportado que se identifica en la estructura de lenguaje. La información de referencia para el asistente XML de CICS lista las restricciones que se aplican a cada lenguaje de alto nivel.

Procedimiento

1. Ejecute el trabajo por lotes DFHLS2SC. DFHLS2SC tiene parámetros opcionales que selecciona para cumplir con los requisitos, como la selección de una página de códigos particular o un espacio de nombres. Utilice los siguientes parámetros como mínimo:
 - a. Especifique el lenguaje de alto nivel de la estructura de lenguaje en el parámetro **LANG**.
 - b. Si está desplegando las correlaciones de datos en un paquete, especifique el nombre de un recurso de paquete en el parámetro **BUNDLE**. Si está creando un enlace XML para un feed Atom, no especifique este parámetro.
 - c. Especifique el nivel de correlación en el parámetro **MAPPING-LEVEL**. Si está creando un enlace XML para un feed Atom, debe utilizar el nivel de correlación de 3.0 o superior. Para otras situaciones, aunque puede utilizar cualquier nivel de correlación, para obtener las opciones de correlación más avanzadas utilice el nivel de correlación más reciente.
 - d. Opcional: Si está creando un enlace XML para un feed Atom, y el registro de datos de aplicación contiene indicaciones de fecha y hora en formato ABSTIME de CICS, especifique el parámetro opcional **DATETIME=PACKED15** para correlacionar estos campos como indicaciones de fecha y hora.

- e. Especifique la ubicación y la página de códigos de las estructuras de lenguaje que describen el registro de aplicación en los parámetros **PDSMEM** y **PDSCP**.
- f. Especifique el nombre y ubicación del archivo de esquema en el parámetro **SCHEMA**. La extensión de archivo es **.xsd**. Si está creando un paquete, no especifique una ubicación. DFHLS2SC crea el esquema XML, pero no la estructura de directorio, si el archivo ya no existe.
- g. Especifique el nombre y la ubicación del esquema XML en el parámetro **XSDBIND**. La extensión de archivo es **.xsdbind**. Si está creando un paquete, no especifique una ubicación. DFHLS2SC crea el enlace XML, pero no la estructura de directorio, si el archivo ya no existe.

Consejo: Ponga el esquema y el enlace XML en la misma estructura de directorio para habilitar la validación. La validación puede ser útil cuando está probando la aplicación en un entorno de prueba o desarrollo. Si está creando un paquete, CICS pone los archivos en el mismo directorio.

Si especifica el parámetro **BUNDLE**, el trabajo por lotes crea una estructura de directorio de paquetes en z/OS UNIX. El directorio de paquetes tiene un subdirectorio **META-INF** que contiene el manifiesto de paquete. El trabajo por lotes crea un esquema XML y un enlace XML en el directorio de paquete, utilizando los nombres de archivo que ha especificado para los parámetros **SCHEMA** y **XSDBIND**. Si no especifica el parámetro **BUNDLE**, el trabajo por lotes crea el esquema XML y el enlace XML sólo en la ubicación especificada.

2. Instale el recurso **BUNDLE** o un recurso **ATOMSERVICE** que especifique este enlace XML. Los recursos **BUNDLE** y **ATOMSERVICE** crean dinámicamente un recurso **XMLTRANSFORM**, que define la ubicación del archivo de enlace y el esquema XML.

Resultados

Cuando genera correlaciones desde estructuras de lenguaje, sólo es posible una transformación de XML.

Ejemplo

El ejemplo siguiente muestra DFHLS2SC con el conjunto mínimo de parámetros especificados.

```
//LS2SC JOB 'accounting information',name,MSGCLASS=A
// SET QT=''
//JAVAPROG EXEC DFHLS2SC,
// TMPFILE=&QT.&SYSUID.&QT
//INPUT.SYSUT1 DD *
LANG=COBOL
BUNDLE=/u/exampleapp/bundle/test1
LOGFILE=/u/exampleapp/xsdbind/example.log
MAPPING-LEVEL=3.0
PDSLIB=//CICSHLQ.SDFHSAMP
PDSMEM=CPYBK2
XSDBIND=example.xsdbind
SCHEMA=example.xsd
/*
```

Qué hacer a continuación

Escriba un programa de aplicación para transformar los datos de aplicación en XML y viceversa. Puede utilizar las mismas correlaciones para ambas

transformaciones. Si ha creado un enlace XML para un feed Atom, continúe con los pasos para configurar el feed Atom.

Generación de correlaciones desde un esquema XML

Para crear datos de aplicación desde XML o viceversa que cumplan con un esquema XML existente, puede crear las correlaciones para describir cómo transformará CICS los datos en tiempo de ejecución. Puede comenzar desde un esquema XML o un documento WSDL.

Antes de empezar

Debe tener un esquema XML válido o un documento WSDL. Antes de crear las correlaciones, debe asegurarse que se han completado estas condiciones previas:

- Debe tener un esquema XML válido o un documento WSDL.
- Debe configurar el ID de usuario bajo el que se ejecuta DFHSC2LS para utilizar los servicios del sistema UNIX.
- El ID de usuario debe tener permiso de lectura para acceder al esquema XML o documento WSDL y escribir el permiso para colocar la salida en los directorios adecuados en z/OS UNIX.
- El ID de usuario debe tener una asignación de almacenamiento lo suficientemente grande para ejecutar Java. Puede utilizar cualquier versión soportada de Java. De forma predeterminada, DFHWS2LS utiliza la versión de Java especificada en el parámetro **JAVADIR**.

Acerca de esta tarea

Utilice el asistente XML de CICS para crear las correlaciones de datos para el esquema XML.

Procedimiento

1. Ejecute el trabajo por lotes DFHSC2LS. DFHSC2LS tiene parámetros opcionales que selecciona para cumplir los requisitos, como la selección de una página de códigos particular o especificar cómo gestionar los datos de caracteres de longitud variable. Utilice los siguientes parámetros como mínimo:
 - a. Especifique la ubicación del archivo de entrada en el parámetro **WSDL** o **SCHEMA**. Puede utilizar un documento WSDL o un esquema XML. Si su archivo de entrada contiene referencias a otros esquemas o documentos en Internet y el sistema utiliza un servidor proxy, especifique el nombre de dominio o dirección IP y el número de puerto del servidor proxy.
 - b. Especifique el lenguaje de alto nivel que desea generar en el parámetro **LANG**. El asistente XML soporta lenguajes COBOL, C, C++ y PL/I.
 - c. Si está desplegando las correlaciones de datos en un paquete, especifique el nombre y la ubicación de un paquete en el parámetro **BUNDLE**.

El asistente XML crea una biblioteca de transformaciones soportadas en el esquema XML. Para cada elemento global del archivo de entrada, el asistente crea una transformación individual.

Si especifica el parámetro **BUNDLE**, el trabajo por lotes crea una estructura de directorio de paquete en z/OS UNIX. El directorio de paquetes tiene un subdirectorio META-INF que contiene el manifiesto de paquete. El trabajo por lotes también crea un enlace XML en un directorio de paquete y coloca las estructuras de lenguaje en la ubicación especificada. El asistente XML también coloca una copia del archivo de entrada en el directorio de paquete. Si no

- especifica el parámetro **BUNDLE**, el trabajo por lotes crea las estructuras de lenguaje y el enlace XML sólo en la ubicación especificada.
2. Instale el recurso **BUNDLE**. El recurso **BUNDLE** crea dinámicamente un recurso **XMLTRANSFORM**, que define la ubicación del esquema XML o documento WSDL, el enlace XML y las estructuras de lenguaje.

Resultados

Al generar correlaciones desde un esquema XML, CICS genera una estructura de lenguaje de cada elemento global presente en el esquema.

Ejemplo

El ejemplo siguiente muestra DFHSC2LS con el conjunto mínimo de parámetros especificados.

```
//SC2LS JOB 'accounting information',name,MSGCLASS=A
// SET QT='''
//JAVAPROG EXEC DFHSC2LS,
// TMPFILE=&QT.&SYSUID.&QT
//INPUT.SYSUT1 DD *
LANG=COBOL
BUNDLE=/u/exampleapp/bundle/test1
LOGFILE=/u/exampleapp/xsdbind/example.log
MAPPING-LEVEL=3.0
PDSLIB=//CICSHLQ.SDFHSAMP
PDSMEM=CPYBK2
XSDBIND=example.xsdbind
SCHEMA=example.xsd
/*
```

Qué hacer a continuación

Escriba un programa de aplicación para transformar los datos de aplicación en XML o viceversa. Puede utilizar las mismas correlaciones para ambas transformaciones.

Transformación de datos de aplicación en XML

Puede escribir un programa de aplicación para transformar datos de aplicación en XML.

Antes de empezar

Debe tener un recurso **XMLTRANSFORM** habilitado que defina el enlace XML y el esquema XML.

Acerca de esta tarea

El asistente XML genera las correlaciones en el enlace XML. Si ha comenzado con una estructura de lenguaje y ha utilizado DFHLS2SC, sólo es posible una transformación para XML. Si ha comenzado con un esquema XML, puede haber muchos XML para transformaciones de estructura de lenguaje por lo que su aplicación debe seleccionar qué elemento XML generar.

Procedimiento

1. El programa de aplicación debe crear un canal y colocar los datos asociados a la estructura de lenguaje en un contenedor de modalidad de bits en ese canal.

2. Utilice el mandato de API **TRANSFORM DATATOXML** para transformar los datos en XML:

```
EXEC CICS TRANSFORM DATATOXML
XMLTRANSFORM('
MyXmlTransformName
')
CHANNEL('
MyChannelName
')
DATCONTAINER('
SourceContainerName
')
XMLCONTAINER('
TargetContainerName
')
```

Si el recurso XMLTRANSFORM soporta sólo una única transformación desde la estructura de lenguaje, no tiene que especificar el tipo de conversión en el mandato. Si son posibles muchas transformaciones, añada las siguientes opciones al programa de aplicación:

```
ELEMNAME(elementName) ELEMNAMELEN(elementNameLength)
```

Estas opciones adicionales indican el elemento XML en el que se van a transformar los datos de aplicación y dónde se van a colocar en el contenedor de salida.

3. Instale el programa de aplicación.

Resultados

Cuando la aplicación ejecuta el mandato **TRANSFORM DATATOXML**, CICS comprueba el recurso XMLTRANSFORM para encontrar las correlaciones en el enlace XML y transforma los datos binarios de aplicación en XML utilizando los contenedores en el canal. El XML se coloca en el contenedor que especifica la aplicación en la opción XMLCONTAINER. El XML se ajusta al esquema XML que se define en el recurso XMLTRANSFORM.

Qué hacer a continuación

También puede utilizar las mismas correlaciones para transformar XML en datos de aplicación. Para obtener detalles, consulte “Transformación de XML en datos de aplicación”.

Transformación de XML en datos de aplicación

Puede escribir un programa de aplicación para transformar XML en datos de aplicación. También puede consultar el XML antes de transformarlo.

Antes de empezar

Debe tener un recurso XMLTRANSFORM habilitado que defina el enlace XML y el esquema XML.

Acerca de esta tarea

El asistente XML de CICS XML genera las correlaciones en el enlace XML. Si ha comenzado con una estructura de lenguaje y ha utilizado DFHLS2SC, sólo es posible una transformación desde XML. Si ha comenzado con un esquema XML, puede haber muchos XML para transformaciones de estructura de lenguaje por lo

que su aplicación debe seleccionar qué estructura de lenguaje utilizar como salida.

Procedimiento

1. Cree un canal y coloque el XML en un contenedor de modo de texto en ese canal. Si la aplicación coloca el XML en un contenedor de modalidad BIT, CICS intenta determinar la codificación de los datos de texto, pero puede tardar más en procesar el contenedor y puede que la codificación no sea correcta.
2. Utilice el mandato de API **TRANSFORM XMLTODATA**. Si sólo es posible una transformación para el XML, puede utilizar el siguiente mandato:

```
EXEC CICS TRANSFORM XMLTODATA
XMLTRANSFORM('
MyXmlTransformName
')
CHANNEL('
MyChannelName
')
XMLCONTAINER('
SourceContainerName
')
DATCONTAINER('
TargetContainerName
')
```

Si es posible hacer más de una transformación, utilice las siguientes opciones adicionales:

```
EXEC CICS TRANSFORM XMLTODATA
XMLTRANSFORM('
MyXmlTransformName
')
CHANNEL('
MyChannelName
')
XMLCONTAINER('
SourceContainerName
')
DATCONTAINER('
TargetContainerName
')
ELEMNAME(
elementName
) ELEMNAMELEN(
elementNameLength
)
```

En el segundo ejemplo, CICS devuelve el nombre del elemento que se ha encontrado en la opción ELEMNAME. La aplicación puede utilizar el nombre del elemento para determinar qué biblioteca de las estructuras de lenguaje utilizar para interpretar el contenido del contenedor de destino.

3. Instale el programa de aplicación.

Resultados

Cuando la aplicación ejecuta el mandato **TRANSFORM XMLTODATA**, CICS utiliza los detalles en los recursos XMLTRANSFORM para transformar XML en datos binarios de aplicación utilizando los contenedores en el canal.

Ejemplo

Qué hacer a continuación

También puede utilizar las mismas correlaciones para transformar datos de aplicación en XML. Para obtener detalles, consulte “Transformación de datos de aplicación en XML” en la página 407.

Correlación y transformación de datos de aplicación y JSON

Puede escribir programas de aplicación para transformar datos binarios de aplicación en JavaScript Object Notation (JSON) y viceversa. CICS soporta varios lenguajes de alto nivel y proporciona un asistente JSON para correlacionar cómo se transforman los datos durante el proceso de tiempo de ejecución. CICS utiliza la misma tecnología para la correlación de datos de aplicación con mensajes JSON, como parte del soporte de servicios web.

Antes de empezar

Debe tener instalado Java para ejecutar el asistente de JSON. Las transformaciones se puede llevar a cabo internamente con CICS o utilizando un servidor JVM. Si utiliza Java para las transformaciones, debe tener instalado un servidor JVM de Axis2 para transformar datos de aplicación y JSON. Consulte Configuring a JVM server for Axis2 para obtener más información.

Acerca de esta tarea

La ventaja de utilizar este enfoque para transformar datos de aplicación a y desde JSON es que CICS va más allá de las prestaciones ofrecidas por un analizador JSON. CICS puede interpretar el JSON y realizar las conversiones basadas en registros de los datos de aplicación. Por lo tanto, es más fácil y más rápido crear aplicaciones que funcionen con JSON utilizando este enfoque.

El asistente de JSON de CICS es un programa de utilidad proporcionado que le ayuda a crear los artefactos de correlación necesarios para transformar datos binarios de aplicación en JSON o para transformar JSON en datos binarios de aplicación. El asistente de JSON crea los artefactos en un directorio de paquete.

Procedimiento

1. Cree un paquete con el asistente de JSON. Este paquete contiene los artefactos de correlación necesarios para la transformación de datos.
2. Instale el paquete en CICS para que las correlaciones estén disponibles.
3. Cree o actualice un programa de aplicación para gestionar la transformación de datos. Dispone de dos opciones:
 - Utilice el mandato **TRANSFORM DATATOJSON** y los mandatos de API **TRANSFORM JSONTODATA** en el programa de aplicación. Es el enfoque recomendado.
 - Utilice el mandato de API **LINK PROGRAM** para enlazar con la interfaz enlazable proporcionada por CICS, DFHJSON.
4. Ejecute la aplicación para probar que la transformación funciona como pretendía.

Resultados

Los datos de aplicación se transforman en JSON o JSON se transforma en datos de aplicación.

Qué hacer a continuación

Los pasos 1 en la página 410 a 4 en la página 410 se explican con más detalles en los temas siguientes.

El asistente de JSON de CICS

El asistente de JSON de CICS es un conjunto de programas de utilidad por lotes que puede ayudarle a generar correlaciones entre estructuras de lenguaje de alto nivel y esquemas JSON para realizar transformaciones entre datos de aplicación y JSON. El asistente soporta un despliegue de aplicaciones rápido que lleva a cabo el procesamiento de JSON con la mínima cantidad de esfuerzo de programación.

Utilizando el asistente JSON para CICS se reduce la cantidad de código que debe escribir para analizar o construir JSON; CICS transforma datos entre fragmentos de JSON y la estructura de datos de un programa de aplicación.

El asistente de JSON puede crear un esquema JSON a partir de una sencilla estructura de lenguaje, o una estructura de lenguaje a partir de un esquema XML existente, y soporta COBOL, C/C++ y PL/I. También genera metadatos que CICS utiliza en el tiempo de ejecución para convertir automáticamente datos XML en datos de aplicación binarios o viceversa; los metadatos se definen en un enlace XML y se almacenan en z/OS UNIX. El esquema para el enlace XML está en el directorio `/usr/lpp/cicsts/cicsts52/schemas/xmltransform/` en z/OS UNIX.

Los programas de utilidad DFHJS2LS y DFHLS2JS se conocen colectivamente como el asistente de JSON de CICS:

DFHLS2JS

DFHLS2JS correlaciona estructuras de lenguaje de alto nivel con esquemas JSON.

DFHJS2LS

DFHJS2LS correlaciona esquemas JSON con estructuras de lenguaje de alto nivel.

Los procedimientos JCL para ejecutar ambos programas están en la biblioteca `hlq.XDFHINST`, donde `hlq` es el calificador de alto nivel de la instalación CICS.

La modalidad de uso relevante para el procedimiento DFHLS2JS o DFHJS2LS depende de los requisitos:

- DFHLS2JS: conversión de lenguaje de alto nivel a esquema JSON para interfaz enlazable
- DFHJS2LS: conversión de esquema JSON a lenguaje de alto nivel para interfaz enlazable
- DFHLS2JS: conversión de lenguaje de alto nivel a esquema JSON para servicios de solicitud-respuesta
- DFHJS2LS: conversión de esquema JSON a lenguaje de alto nivel para servicios solicitud-respuesta
- DFHJS2LS: Conversión de esquema JSON a un lenguaje de alto nivel para servicios RESTful

Las dos correlaciones no son simétricas:

- Si procesa una estructura de datos de lenguaje con DFHLS2JS y, a continuación, procesa el esquema JSON resultante con DFHJS2LS, no espere que la estructura de datos final sea la misma que el original.
- Si procesa un esquema JSON con DFHJS2LS y, a continuación, procesa la estructura de lenguaje resultante con DFHLS2JS, no espere que el esquema JSON final sea el mismo que el original.
- En algunos casos, DFHJS2LS genera estructuras de lenguaje que no soporta DFHLS2JS.

Debe codificar estructuras de lenguaje de alto nivel procesadas por DFHLS2JS según las reglas del lenguaje, como se implementa en los compiladores de lenguaje que soporta CICS.

DFHLS2JS: Conversión de lenguaje de alto nivel a esquema JSON para interfaz vinculable

El procedimiento catalogado de DFHLS2JS genera un esquema JSON y un archivo de enlace JSON a partir de una estructura de lenguaje de alto nivel. Utilice DFHLS2JS cuando desee crear un programa CICS que pueda analizar o crear JSON. Se proporcionan descripciones de las sentencias de control de trabajos, parámetros simbólicos, los parámetros de entrada para DFHLS2JS y un trabajo de ejemplo que utiliza DFHLS2JS para ayudarle a utilizar este procedimiento.

El procedimiento DFHLS2JS JCL está instalado en el conjunto de datos *HLQ* .XDFHINST , donde *HLQ* es el calificador de alto nivel donde está instalado CICS.

Sentencias de control de trabajos para DFHLS2JS

JOB Inicia el trabajo.

EXEC Especifica el nombre del procedimiento (DFHLS2JS).

INPUT.SYSUT1 DD

Especifica la entrada. Los parámetros de entrada se especifican normalmente en el flujo de entrada. Sin embargo, también puede definirlos en un conjunto de datos o en un miembro de conjunto de datos particionados.

Parámetros simbólicos

Los siguientes parámetros simbólicos se definen en DFHLS2JS:

JAVADIR = *vía de acceso*

Especifica el nombre del directorio Java que utiliza DFHLS2JS. El valor de este parámetro se añade a /usr/lpp/ para crear un nombre de vía de acceso completo de /usr/lpp/ *vía de acceso* .

Generalmente, no es necesario especificar este parámetro. El valor predeterminado es el valor proporcionado para el trabajo de instalación (DFHISTAR) de CICS en el parámetro **JAVADIR**.

PATHPREF = *prefijo*

Especifica un prefijo opcional que amplía la vía de acceso de directorio z/OS UNIX que se utiliza en otros parámetros. El valor predeterminado es la serie vacía.

Generalmente, no es necesario especificar este parámetro. El valor predeterminado es el valor proporcionado para el trabajo de instalación (DFHISTAR) de CICS en el parámetro **PATHPREF**.

SERVICE = *valor*

Utilice este parámetro sólo cuando el soporte de IBM le indique que lo haga.

TMPDIR = *tmpdir*

Especifica la ubicación de un directorio en z/OS UNIX que DFHLS2JS utiliza como espacio de trabajo temporal. El ID de usuario bajo el que se ejecuta el trabajo debe tener permiso de lectura y escritura para este directorio.

El valor predeterminado es /tmp.

TMPPFILE = *tmpprefix*

Especifica un prefijo que DFHLS2JS utiliza para construir los nombres de los archivos de espacio de trabajo temporal.

El valor predeterminado es LS2JS.

USSDIR = *vía de acceso*

Especifica el nombre del directorio CICS TS en el sistema de archivos z/OS UNIX. El valor de este parámetro se añade a /usr/lpp/cicsts/ proporcionando un nombre de vía de acceso completo de /usr/lpp/cicsts/*path*. Esto debe especificarse como '.' (punto) si se utiliza el valor predeterminado.

Generalmente, no es necesario especificar este parámetro. El valor predeterminado es el valor proporcionado para el trabajo de instalación (DFHISTAR) de CICS en el parámetro **USSDIR**.

El espacio de trabajo temporal

DFHLS2JS crea estos tres archivos temporales en tiempo de ejecución:

```
tmpdir / tmpprefix .in  
tmpdir / tmpprefix .out  
tmpdir / tmpprefix .err
```

donde:

tmpdir es el valor que se especifica en el parámetro **TMPDIR**.

tmpprefix es el valor que se especifica en el parámetro **TMPPFILE**.

Los nombres predeterminados para los archivos (cuando no se especifica **TMPDIR** y **TMPPFILE** son los siguientes:

```
/tmp/LS2JS.in  
/tmp/LS2JS.out  
/tmp/LS2JS.err
```

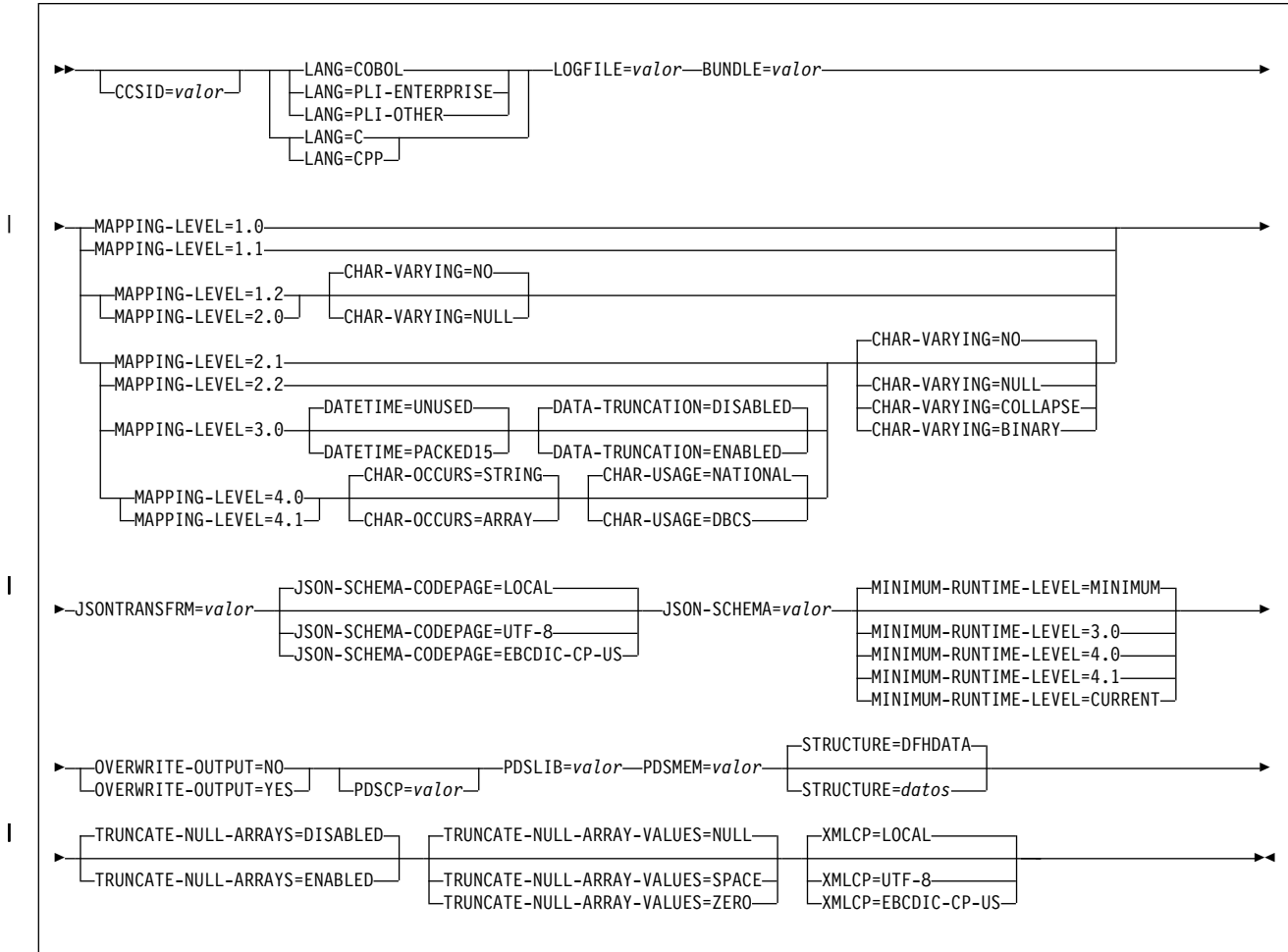
Importante: DFHLS2JS no bloquea el acceso a los archivos z/OS UNIX o los miembros del conjunto de datos. Por lo tanto, si dos o más instancias de DFHLS2JS se ejecutan de forma simultánea y utilizan los mismos archivos de espacio de trabajo temporal, nada impide a un trabajo sobrescribir los archivos de espacio de trabajo mientras otro trabajo los está utilizando, llevando a errores imprevisibles.

Cree un convenio de denominación y procedimientos operativos que eviten esta situación; por ejemplo, puede utilizar el parámetro simbólico de sistema **SYSUID** para generar nombres de archivo de espacios de trabajo que son exclusivos para un usuario individual.

Estos archivos temporales se suprimen antes de la finalización del trabajo.

Parámetros de entrada de DFHLS2JS

El siguiente diagrama de sintaxis muestra los parámetros de entrada disponibles:



Utilización de parámetros

Los parámetros se deben regir por las siguientes reglas:

- Puede especificar los parámetros de entrada en cualquier orden.
- Cada parámetro se debe iniciar en una nueva línea.
- Un parámetro (y su carácter de continuación, si utiliza uno) no deben sobrepasar la columna 72; las columnas 73 a 80 deben contener espacios en blanco.
- Si un parámetro es demasiado largo para ajustarse en una única línea, utilice el carácter de asterisco (*) al final de la línea para indicar que el parámetro continúa en la línea siguiente. Todo (incluidos los espacios) antes del asterisco se considera parte del parámetro.
- Un carácter de signo de almohadilla (#) en la primera posición de carácter de la línea es un carácter de comentario. La línea se ignora.
- Una coma en la última posición de carácter de la línea es un separador de línea opcional y se ignora.

Descripciones de parámetros

BUNDLE = *valor*

Especifica la vía de acceso y el nombre del directorio del paquete en z/OS UNIX. Si especifica este valor, el asistente de JSON genera un paquete que contiene el enlace JSON. La información de vía de acceso para este parámetro sustituye cualquier información de vía de acceso para el parámetro **JSONTRANSFRM**.

Opcionalmente, puede especificar un archivo de archivado en lugar de un nombre de directorio. El asistente de JSON soporta archivos .zip y .jar. Sin embargo, debe descomprimir los archivos antes de intentar instalar el recurso BUNDLE.

CCSID = *valor*

Especifica el CCSID que se utiliza en tiempo de ejecución para codificar datos de caracteres en la estructura de datos de la aplicación. El valor de este parámetro sustituye el valor del parámetro de inicialización del sistema **LOCALCCSID**. El *valor* debe ser un EBCDIC CCSID soportado por los servicios de conversión de Java y z/OS. Si no especifica este parámetro, la estructura de datos de aplicación se codifica utilizando el CCSID especificado en el parámetro de inicialización del sistema.

Puede utilizar este parámetro con cualquier nivel de correlación.

CHAR-VARYING = { **NO** | **NULL** | **COLLAPSE** | **BINARY** }

Especifica cómo los campos de carácter de la estructura de lenguaje se correlacionan cuando el nivel de correlación es 1.2 o superior. Un campo de carácter en COBOL es una cláusula Picture de tipo X; por ejemplo, PIC(X) 10 . Un campo de caracteres en C/C++ es una matriz de caracteres. Este parámetro no se aplica a estructuras de lenguaje Enterprise y Otros PL/I. Puede seleccionar estas opciones:

NO Los campos de carácter se correlacionan con una serie JSON y se procesan como campos de longitud fija. La longitud máxima de los datos es igual a la longitud del campo. NO es el valor predeterminado para el parámetro **CHAR-VARYING** para COBOL y PL/I a niveles de correlación 2.0 y anteriores.

NULL Los campos de carácter se correlacionan con una serie JSON y se procesan como series terminadas en nulo. CICS añade un carácter nulo de terminación cuando se está transformando desde un esquema JSON. La longitud máxima de la serie de caracteres se calcula como un carácter menos que la longitud indicada en la estructura de lenguaje. NULL es el valor predeterminado para el parámetro **CHAR-VARYING** para C/C++.

COLLAPSE

Los campos de caracteres se correlacionan con una serie JSON. Los espacios en blanco finales en este campo no se incluyen en el esquema JSON. COLLAPSE es el valor predeterminado para el parámetro **CHAR-VARYING** para COBOL y PL/I a nivel de correlación 2.1 y en adelante.

BINARY

Los campos de carácter se correlacionan con una serie JSON que contiene datos de codificados en base64 y se procesan como campos de longitud fija. El valor BINARY en el parámetro **CHAR-VARYING** está disponible sólo en niveles de correlación 2.1 y mayores.

CHAR-OCCURS = { STRING | **ARRAY** }

Especifica cómo las matrices de caracteres en la estructura de lenguaje se correlacionan cuando el nivel de correlación es 4.0 o superior. Por ejemplo, PIC X OCCURS 20. Este parámetro sólo lo utiliza el lenguaje COBOL.

ARRAY

Las matrices de caracteres se correlacionan con la matriz JSON. Esto significa que cada carácter se correlaciona como un elemento JSON individual. Este también es el comportamiento a niveles de correlación 3.0 y anteriores.

STRING

Las matrices de caracteres se correlacionan con una serie JSON. Esto significa que toda la matriz COBOL se correlaciona como un único elemento JSON.

CHAR-USAGE = { NATIONAL | **DBCS** }

En COBOL, el tipo de datos nacional, PIC N , se puede utilizar para datos UTF-16 o DBCS. Este valor lo controla la opción del compilador NSYMBOL. Debe establecer el parámetro **CHAR-USAGE** en el asistente en el mismo valor que la opción de compilador NSYMBOL para asegurarse de que los datos se manejan adecuadamente. Normalmente, se establece en CHAR-USAGE=NATIONAL cuando utiliza UTF-16.

DBCS Los datos de los campos PIC (*n*) se tratan como datos cifrados en UTF-16.

NATIONAL

Los datos de los campos PIC (*n*) se tratan como datos cifrados en DBCS.

DATA-TRUNCATION = { DISABLED | **ENABLED** }

Especifica si los datos de longitud variable se toleran en una estructura de campo de longitud fija:

DISABLED

Si los datos son menores que la longitud fija que está esperando CICS, CICS rechaza los datos truncados y emite un mensaje de error.

ENABLED

Si los datos son menores que la longitud fija que está esperando CICS, CICS tolera los datos truncados y procesa los datos que faltan como valores nulos

DATETIME = { UNUSED | **PACKED15** }

Especifica si las propiedades de fecha y hora JSON en la estructura de lenguaje de alto nivel, incluidos los valores ABSTIME de CICS, se correlacionan como indicaciones de fecha y hora:

PACKED15

Cualquier propiedad de fecha y hora de JSON se correlaciona como marca de fecha y hora.

UNUSED

Cualquier propiedad de fecha y hora de JSON no se correlaciona como marca de fecha y hora. Esta correlación es el valor predeterminado.

Este parámetro se puede definir en un nivel de correlación de 3.0.

JSON-SCHEMA = *valor*

El nombre completo del archivo de z/OS UNIX en el que se escribe el esquema JSON.

JSON-SCHEMA-CODEPAGE = { **LOCAL** | **UTF-8** | **EBCDIC-CP-US** }

Especifica la página de códigos que se utiliza para generar los documentos de esquema JSON.

LOCAL

Especifica que los esquemas JSON se generan utilizando la página de códigos predeterminada para el sistema de archivos.

UTF-8 Especifica que los esquemas JSON se generan utilizando la página de códigos UTF-8.

EBCDIC-CP-US

Especifica que los esquemas JSON se generan utilizando la página de códigos US EBCDIC.

JSONTRANSFRM = *valor*

Este parámetro es obligatorio para la modalidad LINKable, pero no es válido para la modalidad de solicitud-respuesta. Indica el nombre que se utiliza para el recurso de paquete **JSONTRANSFRM** en CICS.

LANG = **COBOL**

Especifica que el lenguaje de programación de la estructura de lenguaje de alto nivel es COBOL.

LANG = **PLI-ENTERPRISE**

Especifica que el lenguaje de programación de la estructura de lenguaje de alto nivel es Enterprise PL/I.

LANG = **PLI-OTHER**

Especifica que el lenguaje de programación de la estructura de lenguaje de alto nivel es un nivel de PL/I distinto a Enterprise PL/I.

LANG = **C**

Especifica que el lenguaje de programación de la estructura de lenguaje de alto nivel es C.

LANG = **CPP**

Especifica que el lenguaje de programación de la estructura de lenguaje de alto nivel es C++.

LOGFILE = *value*

El nombre completo del archivo de z/OS UNIX en el que DFHLS2JS escribe la información de rastreo y registro de actividad. Si no existe, DFHLS2JS crea el archivo, pero no la estructura de directorio.

Es posible que la organización de servicio de IBM le solicite utilizar este parámetro si se encuentran problemas con DFHLS2JS.

MAPPING-LEVEL = { **1.0** | **1.1** | **1.2** | **2.0** | **2.1** | **2.2** | **3.0** | **4.0** | **4.1** }

Especifica el nivel de correlación para que el asistente lo utilice al generar las estructuras de lenguaje y enlace JSON. Tiene que utilizar el nivel de correlación 3.0 o posterior.

MINIMUM-RUNTIME-LEVEL = { **MINIMUM** | **3.0** | **4.0** | **4.1 CURRENT** }

Especifica el entorno de tiempo de ejecución de CICS mínimo en el que se puede desplegar el enlace JSON. Si selecciona un nivel que no coincide con los demás parámetros que ha especificado, recibe un mensaje de error. Las opciones que puede seleccionar son las siguientes:

MINIMUM

El nivel de tiempo de ejecución más bajo posible de CICS se asigna automáticamente basándose en los parámetros que ha especificado.

- 3.0 Especifique el nivel de tiempo de ejecución 3.0 o superior si desea utilizar el asistente de JSON de CICS y aprovechar las correlaciones de datos avanzadas.
- 4.0 El archivo de enlace de servicios web generado se despliega correctamente en una región de CICS TS 5.2 o posterior. Con este nivel de ejecución, puede utilizar un nivel de correlación de 4.0 o anterior para el parámetro **MAPPING-LEVEL**. Puede utilizar cualquier parámetro opcional en este nivel.
- 4.1 El archivo de enlace de servicios web generado se despliega correctamente en una región de CICS TS 5.2 que tiene aplicado APAR PI67641, o en cualquier CICS TS 5.3 o región posterior. Con este nivel de ejecución, puede utilizar un nivel de correlación de 4.1 o anterior para el parámetro **MAPPING-LEVEL**.

CURRENT

Utilice este nivel de tiempo de ejecución para desplegar el archivo de enlace generado en una región de CICS que tiene el conjunto de entorno de ejecución establecido en uno utilizado para generar el archivo de enlace.

OVERWRITE-OUTPUT = { NO | YES }

Controle si se pueden sobrescribir los BUNDLE de CICS en el sistema de archivos.

- NO** No se sustituye ningún BUNDLE existente. Si se encuentra un BUNDLE existente, DFHLS2JS emite un mensaje de error DFHPI9689E y finaliza.
- YES** Se sustituye cualquier BUNDLE existente. Si se encuentra un BUNDLE existente, se emite un mensaje DFHPI9683W para informarle que el archivo se ha sustituido.

PDSCP = *valor*

Especifica la página de códigos que se utiliza en los miembros de un conjunto de datos particionados, donde *valor* es un número CCSID o un número de página de códigos Java. Si no especifica este parámetro, se utiliza la página de códigos de los servicios del sistema z/OS UNIX. Por ejemplo, puede especificar PDSCP=037.

PDSLIB = *valor*

Especifica el nombre del conjunto de datos particionados que contiene las estructuras de datos de lenguaje de alto nivel que se van a procesar.

Restricción: Los registros del conjunto de datos particionados debe tener una longitud fija de 80 bytes.

PDSMEM = *valor*

Especifica el nombre del miembro de conjunto de datos particionados que contiene las estructuras de lenguaje de alto nivel que desea procesar.

STRUCTURE = { DFHDATA | *datos* }

Nombre de la estructura de datos de alto nivel en C y C++. El valor predeterminado es DFHDATA.

TRUNCATE-NUL-ARRAYS = { DISABLED | ENABLED }

Especifica cómo se procesan las matrices estructuradas en el nivel de

correlación 4.1 o superior. Si está habilitado, CICS intentará reconocer los registros vacíos dentro de una matriz (consulte TRUNCATE-NULL-ARRAY-VALUES para obtener más información sobre la identificación de registros vacíos). Si se detectan cinco registros de matriz vacíos consecutivos, la matriz se trunca en el primero de esos registros cuando se genera XML/JSON. Esta prestación de recorte sólo se habilita para matrices con contenido estructurado, las matrices de campos primitivos simples no están sujetos al recorte. El recorte de matrices puede dar como resultado una representación más precisa de los datos en JSON/XML, pero no sin riesgo. Si cinco registros de datos consecutivos se identifican de forma incorrecta como almacenamiento no inicializado (quizás porque contienen legítimamente valores bajos), se puede experimentar una pérdida de datos. Si TRUNCATE-NULL-ARRAYS está habilitado y TRUNCATE-NULL-ARRAY-VALUES no se ha establecido, se utiliza el valor predeterminado para TRUNCATE-NULL-ARRAY-VALUES.

TRUNCATE-NULL-ARRAY-VALUES = { NULL | SPACE | ZERO }

Especifica qué valores se tratan como vacíos para el proceso de TRUNCATE-NULL-ARRAYS en el nivel de correlación 4.1 o superior. De forma predeterminada, el valor nulo (0x00, o los valores bajos) se tratan como vacíos. Si todos los bytes de almacenamiento dentro de un registro de una matriz estructurada contienen nulos, entonces todo el registro se considera vacío. Se pueden especificar uno o más de los valores NULL, SPACE y ZERO en una lista separada por comas, donde NULL implica un carácter nulo (0x00), SPACE implica un SBCS EBCDIC Space (0x40) y ZERO implica un cero decimal con zona (0xF0). Cualquier combinación coincidente de los bytes seleccionados dentro un registro de matriz estructurado provocará que todo el registro se identifique como vacío. Si TRUNCATE-NULL-ARRAY-VALUES tiene un valor definido, entonces TRUNCATE-NULL-ARRAYS debe estar habilitado.

```
//LS2JS JOB '
información de contabilidad
',
name,MSGCLASS=A
// SET QT='''
//JAVAPROG EXEC DFHLS2JS,
// TMPFILE=&QT.&SYSUID.&QT
/INPUT.SYSUT1 DD *
PDSLIB=//CICSHLQ.SDFHSAMP
PDSMEM=CPYBK2
JSON-SCHEMA=example.json
LANG=COBOL
LOGFILE=/u/exampleapp/example.log
MAPPING-LEVEL=4.0
JSONTRANSFRM=EXAMPLE
BUNDLE=/u/exampleapp/bundles/exampleBundle
/*
```

DFHJS2LS: Conversión de esquema JSON a un lenguaje de alto nivel para interfaz enlazable

El procedimiento catalogado de DFHJS2LS genera una estructura de datos de lenguaje de alto nivel y un archivo de enlace de servicio JSON a partir de un esquema JSON. Utilice DFHJS2LS cuando desee crear un programa CICS que pueda analizar o crear JSON. Este tema lista las sentencias de control de trabajo, parámetros simbólicos, parámetros de entrada y sus descripciones DFHJS2LS.

El procedimiento DFHJS2LS JCL está instalado en el conjunto de datos *HLQ*.XDFHINST, donde *HLQ* es el calificador de alto nivel donde está instalado CICS.

Sentencias de control de trabajos para DFHJS2LS

JOB Inicia el trabajo.

EXEC Especifica el nombre del procedimiento (DFHJS2LS).

INPUT.SYSUT1 DD

Especifica la entrada. Los parámetros de entrada se especifican normalmente en el flujo de entrada. También puede definirlos en un conjunto de datos o en un miembro de un conjunto de datos particionados.

Parámetros simbólicos

Los siguientes parámetros simbólicos se definen en DFHJS2LS:

JAVADIR = *vía de acceso*

Especifica el nombre del directorio Java que utiliza DFHJS2LS. El valor de este parámetro se añade a /usr/lpp/ para crear un nombre de vía de acceso completo de /usr/lpp/ *vía de acceso* .

Generalmente, no es necesario especificar este parámetro. El valor predeterminado es el valor proporcionado para el trabajo de instalación (DFHISTAR) de CICS en el parámetro **JAVADIR**.

PATHPREF = *prefijo*

Especifica un prefijo opcional que amplía la vía de acceso de directorio z/OS UNIX que se utiliza en otros parámetros. El valor predeterminado es la serie vacía.

Generalmente, no es necesario especificar este parámetro. El valor predeterminado es el valor proporcionado para el trabajo de instalación (DFHISTAR) de CICS en el parámetro **PATHPREF**.

SERVICE = *valor*

Utilice este parámetro sólo cuando el soporte de IBM le indique que lo haga.

TMPDIR = *tmpdir*

Especifica la ubicación de un directorio en z/OS UNIX que DFHJS2LS utiliza como espacio de trabajo temporal. El ID de usuario bajo el que se ejecuta el trabajo debe tener permiso de lectura y escritura para este directorio.

El valor predeterminado es /tmp.

TMPFILE = *tmpprefix*

Especifica un prefijo que DFHJS2LS utiliza para construir los nombres de los archivos de espacio de trabajo temporal.

El valor predeterminado es JS2LS.

USSDIR = *vía de acceso*

Especifica el nombre del directorio CICS TS en el sistema de archivos de los servicios del sistema UNIX. El valor de este parámetro se añade a /usr/lpp/cicsts/ para crear un nombre de vía de acceso completo de /usr/lpp/cicsts/ *vía de acceso* . Esto debe especificarse como '.' (punto) si se utiliza el valor predeterminado.

Generalmente, no es necesario especificar este parámetro. El valor predeterminado es el valor proporcionado para el trabajo de instalación (DFHISTAR) de CICS en el parámetro **USSDIR**.

El espacio de trabajo temporal

DFHJS2LS crea estos tres archivos temporales en tiempo de ejecución:

```
tmpdir / tmpprefix .in  
tmpdir / tmpprefix .out
```

tmpdir / tmpprefix .err

donde:

tmpdir es el valor que se especifica en el parámetro **TMPDIR**.

tmpprefix es el valor que se especifica en el parámetro **TMPFILE**.

Los nombres predeterminados para los archivos (cuando no se especifica **TMPDIR** y **TMPFILE** son los siguientes:

/tmp/JS2LS.in

/tmp/JS2LS.out

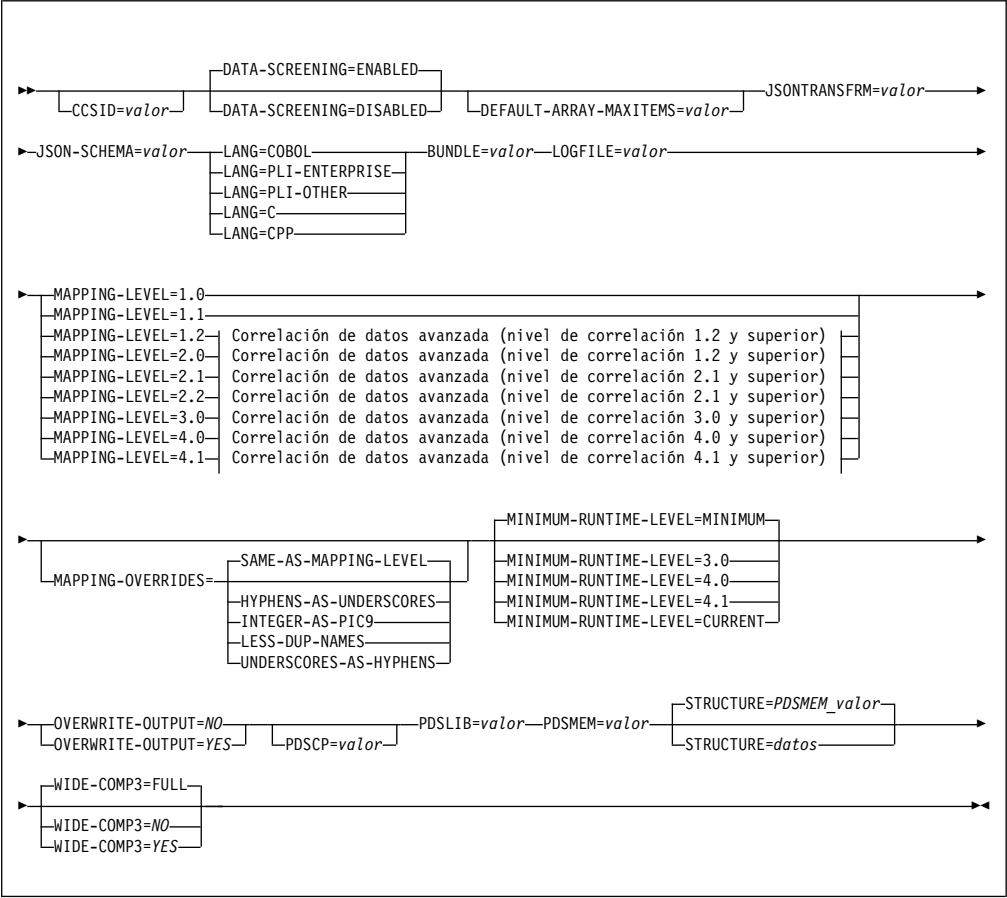
/tmp/JS2LS.err

Importante: DFHJS2LS no bloquea el acceso a los archivos z/OS UNIX o los miembros del conjunto de datos. Por lo tanto, si dos o más instancias de DFHJS2LS se ejecutan de forma simultánea y utilizan los mismos archivos de espacio de trabajo temporal, nada impide a un trabajo sobrescribir los archivos de espacio de trabajo mientras otro trabajo los está utilizando, llevando a errores imprevisibles.

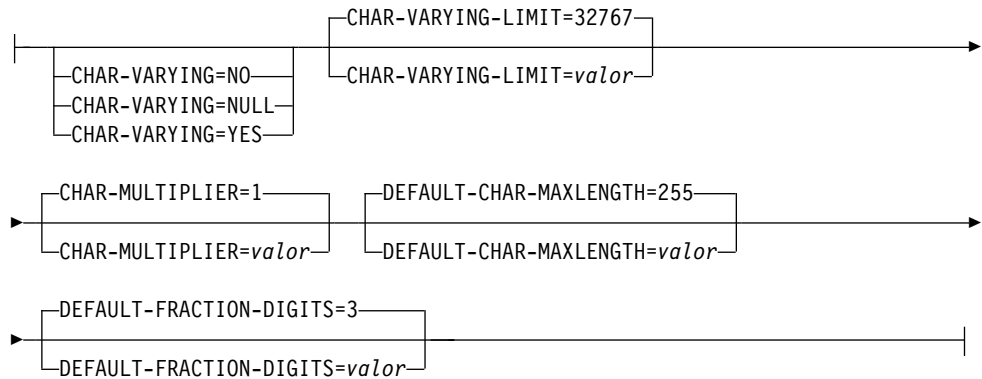
Por lo tanto, se aconseja crear un convenio de denominación y procedimientos operativos que eviten esta situación; por ejemplo, puede utilizar el parámetro simbólico de sistema **SYSUID** para generar nombres de archivo de espacios de trabajo que son exclusivos para un usuario individual.

Estos archivos temporales se suprimen antes de la finalización del trabajo.

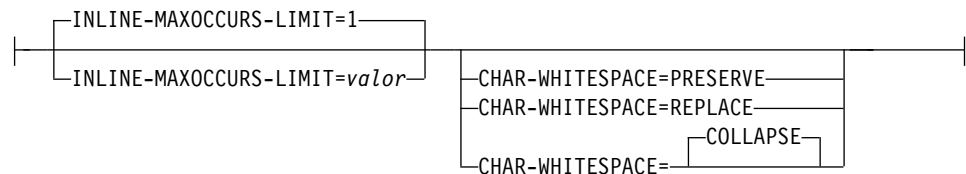
Parámetros de entrada de DFHJS2LS



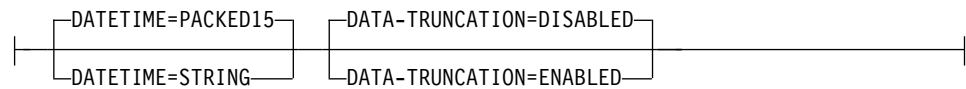
Correlación de datos avanzada (nivel de correlación 1.2 y superior):



Correlación de datos avanzada (nivel de correlación 2.1 y superior):



Correlación de datos avanzada (nivel de correlación 3.0 y superior):



Utilización de parámetros

- Puede especificar los parámetros de entrada en cualquier orden.
- Cada parámetro se debe iniciar en una nueva línea.
- Un parámetro (y su carácter de continuación, si utiliza uno) no deben sobrepasar la columna 72; las columnas 73 a 80 deben contener espacios en blanco.
- Si un parámetro es demasiado largo para ajustarse en una única línea, utilice el carácter de asterisco (*) al final de la línea para indicar que el parámetro continúa en la línea siguiente. Todo (incluidos los espacios) antes del asterisco se considera parte del parámetro.
- Un carácter # en la primera posición de carácter de la línea es un carácter de comentario. La línea se ignora.
- Una coma en la última posición de carácter de la línea es un separador de línea opcional y se ignora.

Descripciones de parámetros

BUNDLE = *valor*

Especifica la vía de acceso y el nombre del directorio del paquete en z/OS UNIX. Si especifica este valor, el asistente de JSON genera el enlace JSON en el

directorio de paquete y crea un manifiesto de paquete. La información de vía de acceso para este parámetro sustituye cualquier información de vía de acceso para el parámetro **JSONTRANSFRM**.

CCSID = *valor*

Especifica el CCSID que se utiliza en tiempo de ejecución para codificar datos de caracteres en la estructura de datos de la aplicación. El valor de este parámetro sustituye el valor del parámetro de inicialización del sistema **LOCALCCSID**. El *valor* debe ser un EBCDIC CCSID soportado por los servicios de conversión de Java y z/OS. Si no especifica este parámetro, la estructura de datos de aplicación se codifica utilizando el CCSID especificado en el parámetro de inicialización del sistema.

Puede utilizar este parámetro con cualquier nivel de correlación.

CHAR-MULTIPLIER = { 1 | *valor* }

Especifica el número de bytes que se permiten para cada carácter cuando el nivel de correlación es 1.2 o posterior. El *valor* de este parámetro puede ser un entero positivo en el rango de 1 a 2.147.483.647. Todas las correlaciones basadas en caracteres no numéricos están sujetas a este multiplicador. Los campos binario, numérico, con zona y decimal empaquetado no están sujetos a este multiplicador.

Este parámetro puede ser útil si, por ejemplo, está planificando utilizar caracteres DBCS donde puede optar por un multiplicador de 3 para permitir espacio para caracteres de cambio y desplazamiento potenciales alrededor de todos los caracteres de doble byte en tiempo de ejecución.

Cuando establece **CCSID=1200** (indicando UTF-16), los únicos valores válidos para **CHAR-MULTIPLIER** son 2 o 4. Cuando utiliza UTF-16, el valor predeterminado es 2. Utilice **CHAR-MULTIPLIER=2** cuando espera que los datos de aplicación contengan caracteres que requieren 1 unidad de codificación UTF-16. Utilice **CHAR-MULTIPLIER=4** cuando espera que los datos de aplicación contengan caracteres que requieren 2 unidades de codificación UTF-16.

Nota: Establecer **CHAR-MULTIPLIER** en 1 no excluye el uso de caracteres DBCS, y establecerlo en 2 no excluye el uso de pares de sustitución UTF-16. Sin embargo, si los caracteres amplios utilizan de forma rutinaria algunos valores válidos no se ajustarán al campo asignado. Si se utiliza un valor **CHAR-MULTIPLIER** mayor, es posible almacenar más caracteres en el campo asignado que los que son válidos en el XML. Se debe tener cuidado para ajustarse a las restricciones de rango adecuadas.

CHAR-VARYING = { **NO** | **NULL** | **YES** }

Especifica cómo se correlacionan datos de caracteres de longitud variable cuando el nivel de correlación es 1.2 o superior. Los tipos de datos binarios de longitud variable se correlacionan siempre con un contenedor o una estructura variable. Si no especifica este parámetro, la correlación predeterminada depende del lenguaje especificado. Puede seleccionar estas opciones:

NO Los datos de caracteres de longitud variable se correlacionan como series de longitud fija.

NULL Los datos de caracteres de longitud variable se correlacionan con series terminadas en nulo.

YES Los datos de caracteres de longitud variable se correlacionan con un tipo de datos CHAR VARYING en PL/I. En los lenguajes COBOL, C y C++, los datos de caracteres de longitud variable se correlacionan con

una representación equivalente que consta de dos elementos relacionados: la longitud de datos y los datos.

CHAR-VARYING-LIMIT = { 32767 | *valor* }

Especifica el tamaño máximo de datos binarios y de datos de caracteres de longitud variable que se correlacionan con la estructura de lenguaje cuando el nivel de correlación es 1.2 o superior. Si los datos binarios o de caracteres son mayores que el valor especificado en este parámetro, se correlacionan con un contenedor y el nombre de contenedor se utiliza en la estructura de lenguaje generada. El *valor* puede ir de 0 al valor predeterminado de 32.767 bytes.

CHAR-WHITESPACE = COLLAPSE | REPLACE | PRESERVE

Especifica cómo CICS manejará el espacio en blanco en los valores de tipo cadena.

COLLAPSE

Los espacios en blanco de inicio, final e incluidos se eliminan y todas las pestañas, líneas nuevas y espacios consecutivos se sustituyen por caracteres de un único espacio.

REPLACE

Cualquier pestaña o línea nueva se sustituye por el número de espacios adecuado.

PRESERVE

Retiene cualquier espacio en blanco en el valor de datos.

Si el parámetro **CHAR-WHITESPACE** no está establecido, el espacio en blanco se contraerá.

Nota: Este parámetro no se aplica a campos con un formato de date-time , uri , base64Binary o hexBinary , donde el espacio en blanco siempre se contrae.

DATA-SCREENING = { ENABLED | DISABLED }

Especifica si los datos proporcionados por la aplicación se analizan para detectar errores.

ENABLED

Los datos de tiempo de ejecución proporcionados por la aplicación que son incoherentes con la estructura de lenguaje, se tratan como un error y se emite un mensaje DFHPI1010. Se devuelve una respuesta de error a la aplicación.

DISABLED

Los valores de datos de los datos de tiempo de ejecución proporcionados por la aplicación que son incoherentes con la estructura de lenguaje se sustituyen por valores predeterminados. Por ejemplo, un cero sustituye un valor erróneo en un campo numérico. No se emite mensaje DFHPI1010 y se devuelve una respuesta normal a la aplicación. Esta característica se puede utilizar para evitar las respuestas de error INVALID_PACKED_DEC y INVALID_ZONED_DEC generadas desde los campos de salida no inicializados.

DATA-TRUNCATION = { DISABLED | ENABLED }

Especifica si los datos de longitud variable se toleran en una estructura de campo de longitud fija:

DISABLED

Si los datos son menores que la longitud fija que está esperando CICS, CICS rechaza los datos truncados y emite un mensaje de error.

ENABLED

Si los datos son menores que la longitud fija que está esperando CICS, CICS tolera los datos truncados y procesa los datos que faltan como valores nulos

DATETIME = { **PACKED15** | **STRING** }

Especifica que las propiedades de fecha y hora JSON se correlacionan con el formato de datos ABSTIME de CICS o con el texto:

PACKED15

Los campos de propiedades de fecha y hora JSON se correlacionan con el formato ABSTIME de CICS.

STRING

Las propiedades de fecha y hora de JSON se correlacionan con texto. Esta correlación es la misma que todos los niveles de correlación anteriores.

Este parámetro se puede utilizar en un nivel de correlación de 3.0.

DEFAULT-ARRAY-MAXITEMS = *valor*

Especifica el límite de matriz máximo a aplicar donde no se implica información de aparición máxima (maxItems) en el esquema JSON. Si no se establece este parámetro, no se aplica límite máximo. El valor de este parámetro puede ser un entero positivo en el rango de 1 a 2147483647. Este parámetro se puede combinar con el uso del parámetro **INLINE-MAXOCCURS-LIMIT** para que influya en la forma en la que se correlacionan las matrices JSON en las estructuras de lenguaje.

DEFAULT-CHAR-MAXLENGTH = { **255** | *valor* }

Especifica la longitud de matriz predeterminada de los datos de caracteres en caracteres para correlaciones donde no está implicada la longitud en el documento de esquema JSON, cuando el nivel de correlación es 1.2 o posterior. El *valor* de este parámetro puede ser un entero positivo en el rango de 1 a 2 147 483 647.

DEFAULT-FRACTION-DIGITS = { **3** | *valor* }

Especifica el número de dígitos de fracción predeterminado a utilizar en un tipo de esquema decimal XML. El valor predeterminado es 3. Para COBOL, el rango válido es 0-17, o 0-30 si se está utilizando **WIDE-COMP3**. Para C o PLI, el rango válido es 0-30.

INLINE-MAXOCCURS-LIMIT = { **1** | *valor* }

Especifica si se utiliza el contenido de repetición variable en línea basado en el atributo maxItems de las palabras clave del esquema JSON.

El parámetro **INLINE-MAXOCCURS-LIMIT** está disponible sólo del nivel de correlación 2.1 en adelante. El *valor* de **INLINE-MAXOCCURS-LIMIT** puede ser un entero positivo en el rango de 0 a 32 767. Un valor de 0 indica que no se utiliza la correlación en línea. Un valor de 1 garantiza que los elementos opcionales se correlacionan en línea. Si el *valor* del atributo maxItems es mayor que el *valor* de **INLINE-MAXOCCURS-LIMIT**, se utiliza la correlación basada en contenedor; de lo contrario, se utiliza la correlación en línea.

Si decide que desea que las listas de repetición variable se correlacionen en línea, tenga en cuenta la longitud de un único elemento de datos recurrentes. Si tiene algunas instancias de longitud larga, es preferible la correlación basada en contenedor, si tiene muchas instancias de longitud corta, es probable que se prefiera la correlación en línea.

JSONTRANSFRM = *valor*

Este parámetro es obligatorio para la modalidad LINKable, pero no es válido para las modalidades de solicitud-respuesta y RESTful. Indica el nombre que se utiliza para el recurso de paquete **JSONTRANSFRM** en CICS.

JSON-SCHEMA = *valor*

El nombre completo del archivo de z/OS UNIX desde el que se lee el esquema JSON. Si no existe, DFHJS2LS crea el archivo, pero no la estructura de directorio.

LANG = **COBOL**

Especifica que el lenguaje de programación de la estructura de lenguaje de alto nivel es COBOL.

LANG = **PLI-ENTERPRISE**

Especifica que el lenguaje de programación de la estructura de lenguaje de alto nivel es Enterprise PL/I.

LANG = **PLI-OTHER**

Especifica que el lenguaje de programación de la estructura de lenguaje de alto nivel es un nivel de PL/I distinto a Enterprise PL/I.

LANG = **C**

Especifica que el lenguaje de programación de la estructura de lenguaje de alto nivel es C.

LANG = **CPP**

Especifica que el lenguaje de programación de la estructura de lenguaje de alto nivel es C++.

LOGFILE = *valor*

El nombre completo del archivo de z/OS UNIX en el que DFHJS2LS escribe la información de rastreo y registro de actividad. Si no existe, DFHJS2LS crea el archivo, pero no la estructura de directorio.

Normalmente, no utiliza este archivo, pero puede solicitarlo la organización de servicio de IBM si encuentra problemas con DFHJS2LS.

MAPPING-LEVEL = { **1.0** | **1.1** | **1.2** | **2.0** | **2.1** | **2.2** | **3.0** | **4.0** | **4.1** }

Especifica el nivel de correlación para que el asistente lo utilice al generar las estructuras de lenguaje y enlace JSON. Debe utilizar el nivel de correlación más reciente disponible; para DFHJS2LS, debe utilizar un nivel de correlación de 3.0 o superior.

3.0 Es el nivel de correlación mínimo que puede utilizar con DFHJS2LS.

4.0 Utilice este nivel de correlación con una región de CICS TS 5.2 cuando dese utilizar UTF-16.

4.1 Para un soporte de matriz truncable, utilice este nivel de correlación con una región de CICS TS 5.2 que tiene aplicado APAR PI67641, o cualquier región de CICS TS 5.3 o posterior.

MAPPING-OVERRIDES = { **SAME-AS-MAPPING-LEVEL** **HYPHENS-AS-UNDERSCORES** | | **INTEGER-AS-PIC9** | **LESS-DUP-NAMES** | **UNDERSCORES-AS-HYPHENS** }

Especifica si el comportamiento predeterminado se sustituye para el nivel de correlación especificado cuando se generan estructuras de lenguaje.

SAME-AS-MAPPING-LEVEL

Este parámetro genera estructuras de lenguaje en el mismo estilo que el nivel de correlación. Es el valor predeterminado.

HYPHENS-AS-UNDERSCORES

Sólo para PL/I. Este parámetro convierte cualquier guión del esquema JSON en guiones bajos en lugar del carácter X, para mejorar la legibilidad de las estructuras de lenguaje PL/I generadas. Para obtener más información, consulte el apartado JSON schema to PL/I mapping.

INTEGER-AS-PIC9

Únicamente para COBOL y DFHJS2LS. Este parámetro genera estructuras de lenguaje que contienen valores enteros del esquema JSON como numerales en lugar de caracteres alfanuméricos.

LESS-DUP-NAMES

Este parámetro genera nombres de campo de estructura no estructural con `_value` al final del nombre para habilitar la referencia directa al campo. Por ejemplo, en la siguiente estructura PLI, cuando se especifica `MAPPING-OVERRIDES=LESS-DUP-NAMES`, el campo de nivel 12 `streetName` lleva el sufijo `_value` :

```
09 streetName,  
12 streetName CHAR(255) VARYING  
UNALIGNED,  
12 filler BIT (7),  
12 attr_nil_streetName_value BIT (1),
```

La estructura resultante es la siguiente:

```
09 streetName,  
12 streetName_value CHAR(255) VARYING  
UNALIGNED,  
12 filler BIT (7),  
12 attr_nil_streetName_value BIT (1),
```

UNDERSCORES-AS-HYPHENS

Esta opción está habilitada automáticamente en el nivel de correlación 4.0.

Solo para COBOL. Este parámetro convierte cualquier guión bajo en esquema JSON en guiones, en lugar del carácter X, para mejorar la legibilidad de las estructuras de lenguaje COBOL generadas. Si se produce un conflicto de nombres de campo, los campos se numeran para garantizar que con exclusivos. Para obtener más información, consulte el apartado JSON schema to COBOL mapping.

MINIMUM-RUNTIME-LEVEL = { MINIMUM | 3.0 | 4.0 | 4.1 | CURRENT }

Especifica el entorno de tiempo de ejecución de CICS mínimo en el que se puede desplegar el enlace JSON. Si selecciona un nivel que no coincide con los demás parámetros que ha especificado, recibe un mensaje de error. Las opciones que puede seleccionar son las siguientes:

MINIMUM

El nivel de tiempo de ejecución más bajo posible de CICS se asigna automáticamente basándose en los parámetros que ha especificado.

3.0 Especifique el nivel de tiempo de ejecución 3.0 o superior si desea utilizar el asistente de JSON de CICS y aprovechar las correlaciones de datos avanzadas.

4.0 El archivo de enlace de servicios web generado se despliega correctamente en una región de CICS TS 5.2 o posterior. Con este nivel de ejecución, puede utilizar un nivel de correlación de 4.0 o anterior para el parámetro **MAPPING-LEVEL**. Puede utilizar cualquier parámetro opcional en este nivel.

- 4.1 El archivo de enlace de servicios web generado se despliega correctamente en una región de CICS TS 5.2 que tiene aplicado APAR PI67641, o en cualquier CICS TS 5.3 o región posterior. Con este nivel de ejecución, puede utilizar un nivel de correlación de 4.1 o anterior para el parámetro MAPPING-LEVEL.

CURRENT

Utilice este nivel de tiempo de ejecución para desplegar el enlace JSON en una región de CICS que tiene el mismo entorno de tiempo de ejecución que la región utilizada para generar el enlace JSON.

OVERWRITE-OUTPUT = { NO | YES }

Controle si se pueden sobrescribir los BUNDLE de CICS en el sistema de archivos.

NO No se sustituye ningún BUNDLE existente. Si se encuentra un BUNDLE existente, DFHJS2LS emite un mensaje de error DFHPI9689E y finaliza.

YES Se sustituye cualquier BUNDLE existente. Si se encuentra un BUNDLE existente, se emite un mensaje DFHPI9683W para informarle que el archivo se ha sustituido.

PDSCP = *valor*

Especifica la página de códigos que se utiliza en los miembros de un conjunto de datos particionados, donde *valor* es un número CCSID o un número de página de códigos Java. Si no especifica este parámetro, se utiliza la página de códigos de los servicios del sistema z/OS UNIX. Por ejemplo, puede especificar PDSCP=037.

PDSLIB = *valor*

Especifica el nombre del conjunto de datos particionados que contiene el lenguaje de alto nivel generado.

PDSMEM = *valor*

Especifica el prefijo de 1 a 6 caracteres que utiliza DFHJS2LS para generar el nombre del miembro del conjunto de datos particionados que contendrá las estructuras de lenguaje de alto nivel.

DFHJS2LS genera un miembro de conjunto de datos particionados para cada operación. Genera el nombre de miembro añadiendo un número de dos dígitos al prefijo.

STRUCTURE = { PDSMEM_value | *datos* }

Nombre de la estructura de datos de alto nivel en C y C++. El valor predeterminado es el valor del parámetro **PDSMEM**.

WIDE-COMP3 = { FULL | NO | YES }

Controla el tamaño máximo de la longitud variable decimal empaquetada en la estructura de lenguaje COBOL o PL/I generada.

FULL Para COBOL y PL/I. DFHJS2LS genera un campo decimal empaquetado que es lo suficientemente grande como para contener todos los valores válidos. El tamaño máximo es de 31 dígitos. Es el valor predeterminado.

NO Solo para COBOL. DFHJS2LS limita la longitud variable de decimal empaquetado a 10 al generar el tipo de estructura de lenguaje COBOL COMP-3. Si el tamaño del decimal empaquetado es mayor de 18, se emite un mensaje DFHPI9022W para indicar que el tipo especificado se está restringiendo a un total de 18 dígitos.

YES Solo para COBOL. DFHJS2LS soporta el tamaño máximo de 31 al generar el tipo de estructura de lenguaje COBOL COMP-3.

Nota: Las opciones NO y YES generan campos que no pueden representar todos los valores válidos; la opción FULL evita este problema. Sin embargo, la opción FULL no permite que algunos valores no válidos se representen en el campo de decimal empaquetado. Por ejemplo, si un esquema indica que hay un máximo de cinco dígitos y un máximo de dos dígitos fraccionales, la opción FULL generará un campo de decimal empaquetado que permite siete dígitos, y esto permite espacio para valores válidos como 25000 y 999,99, pero también proporciona espacio para algunos valores no válidos como 9999,99. Cuando utilice la opción FULL, tenga cuidado de no generar valores no válidos en datos de aplicación.

Ejemplo

```
//JS2LS JOB '  
información de contabilidad  
'  
,  
name,MSGCLASS=A  
// SET QT=''''  
//JAVAPROG EXEC DFHJS2LS,  
// TMPFILE=&QT.&SYSUID.&QT  
/INPUT.SYSUT1 DD *  
PDSLIB=//CICSHLQ.SDFHSAMP  
PDSMEM=CPYBK2  
JSON-SCHEMA=example.json  
LANG=COBOL  
LOGFILE=/u/exampleapp/example.log  
MAPPING-LEVEL=4.0  
CHAR-VARYING=NULL  
JSONTRANSFORM=EXAMPLE  
BUNDLE=/u/exampleapp/bundles/exampleBundle  
/*
```

Niveles de correlación para el asistente de JSON de CICS

Una correlación es un conjunto de reglas que especifica cómo se convierte la información entre estructuras de lenguajes y esquemas JSON. Para beneficiarse de las correlaciones más sofisticadas disponibles, se recomienda establecer el parámetro **MAPPING-LEVEL** en el asistente de CICS en el nivel más reciente.

Cada nivel de correlación hereda la función de la correlación anterior, con el nivel de correlación más alto ofreciendo las mejores prestaciones disponibles. El nivel de correlación más alto proporciona más control sobre la conversión de datos en tiempo de ejecución y elimina restricciones en el soporte para determinados tipos de datos y propiedades JSON.

Puede establecer el parámetro **MAPPING-LEVEL** en un nivel anterior si desea volver a desplegar las aplicaciones que se habían habilitado previamente en ese nivel.

Limitaciones en todos los niveles de correlación

- Los tipos de datos utilizados en el esquema JSON deben declararse de forma explícita.
- El objeto JSON hace referencia a documentos externos utilizando \$ref que no se soportan en el esquema JSON.

Nivel de correlación 4.1

El nivel de correlación 4.1 es compatible con una región CICS TS 5.3 con APAR PI67641 aplicado y posterior.

El nivel de correlación 4.1 se añade a los asistentes de servicios web, los asistentes XML y asistentes de JSON. Este nivel de correlación implementa correlaciones mejoradas para matrices sencillas generadas de abajo arriba a partir de libros de copias existentes; también añade la capacidad de CICS para detectar automáticamente esos registros a partir del formato XML/JSON generado.

DFHLS2WS, DFHWS2LS, DFHLS2SC, DFHSC2LS, DFHJS2LS y DFHLS2JS soportan los parámetros **TRUNCATE=NULL-ARRAYS** y **TRUNCATE=NULL-ARRAY-VALUES**.

Si especifica cualquier valor para **TRUNCATE=NULL-ARRAY-VALUES**, también debe especificar **TRUNCATE=NULL-ARRAYS=ENABLED**.

Nivel de correlación 4.0

Este nivel de correlación ofrece las siguientes compatibilidades:

En el nivel de correlación 4.0 y superior, DFHLS2JS soporta la cláusula COBOL OCCURS DEPENDING ON y la correlación de matrices de caracteres COBOL en esquemas JSON. Puede establecer este comportamiento utilizando el parámetro **CHAR-OCCURS** en el asistente de JSON de CICS.

- Debe especificar el parámetro **DATA-TRUNCATION=ENABLED**.
- Las cláusulas OCCURS DEPENDING ON complejas no son compatibles. Esta limitación significa que OCCURS DEPENDING ON solo es válida para el último campo de una estructura.
- CICS no soporta nombres completos (que utilizan la palabra clave 'OF') como destino de una cláusula OCCURS DEPENDING ON, por ejemplo FIELD1 OF STRUCTURE1.
- CICS no soporta la palabra clave UNBOUNDED. Debe especificar un entero enlazado para el tamaño máximo de la tabla esperada por la aplicación.

En el nivel de correlación 4.0 y superior, los servicios web JSON soportan la conversión de datos de aplicación que se codifican utilizando UTF-16 Unicode.

- Cuando utiliza LS2JS, puede habilitar este comportamiento utilizando tipos de datos específicos del lenguaje para UTF-16.
- Cuando utiliza JS2LS, puede habilitar este comportamiento estableciendo CCSID=1200.
- CICS soporta únicamente una única página de códigos Unicode, " UTF-16BE con IBM Private Use Area " (CCSID 1200).
- No se soporta la conversión de datos de aplicación que se codifican utilizando UTF-8.

Nota: DFHLS2JS no soporta la cláusula COBOL GROUP USAGE NATIONAL.

Nivel de correlación 3.0 y posterior

Este nivel de correlación ofrece las siguientes compatibilidades:

- DFHJS2LS correlaciona tipos de datos date-time con el formato CICS ASKTIME.

- DFHLS2JS puede generar un esquema JSON y un enlace de servicio web desde una aplicación que utiliza muchos contenedores en lugar de sólo un contenedor.
- Toleración de datos truncados descrita por una estructura de datos de longitud fija. Puede establecer este comportamiento utilizando el parámetro **DATA-TRUNCATION** en el asistente de CICS.

Nivel de correlación 2.2 y posterior

Esta opción se retiene para la compatibilidad con servicios web SOAP. No se recomienda utilizar con JSON.

Nivel de correlación 2.1 y posterior

Esta opción se retiene para la compatibilidad con servicios web SOAP. No se recomienda utilizar con JSON.

Este nivel de correlación incluye un mayor control sobre la forma en la que se maneja el contenido de la variable con el nuevo parámetro **INLINE-MAXOCCURS-LIMIT** y los nuevos valores en el parámetro **CHAR-VARYING**.

En el nivel de correlación 2.1 y superior, DFHJS2LS ofrece el siguiente soporte nuevo y mejorado para matrices:

- El parámetro **INLINE-MAXOCCURS-LIMIT**
- La propiedad `minItems`

El parámetro **INLINE-MAXOCCURS-LIMIT** especifica si las listas de repetición variable se correlacionan en línea. Para obtener más información sobre la correlación del contenido que se repite de forma variable en línea, consulte “Matrices de variables de elementos en DFHJS2LS” en la página 283.

En el nivel de correlación 2.1 y superior, DFHLS2JS soporta las siguientes correlaciones JSON:

- No se tienen en cuenta los campos FILLER en COBOL y PL/I
- Un valor de COLLAPSE para el parámetro **CHAR-VARYING**
- Un valor de BINARY para el parámetro **CHAR-VARYING**

Los campos FILLER en COBOL y PL/I se ignoran; no aparecen en el esquema JSON generado y queda un hueco adecuado en las estructuras de datos en el tiempo de ejecución.

COLLAPSE hace que CICS ignore los espacios finales en los campos de texto.

BINARY proporciona soporte para los campos binarios. Este valor es útil al convertir COBOL en un esquema JSON. Esta opción está disponible sólo en matrices de caracteres SBCS y permite que la matriz se correlacione con una serie JSON de longitud fija que contiene datos codificados en base64 en lugar de una serie normal.

Nivel de correlación 1.2 y superior

Esta opción se retiene para la compatibilidad con servicios web SOAP. No se recomienda utilizar con JSON.

Hay disponible un mayor control sobre la manera en la que se transforman los datos binarios y los caracteres en el tiempo de ejecución con estos parámetros adicionales en las herramientas por lotes:

- **CHAR-VARYING**
- **CHAR-VARYING-LIMIT**
- **CHAR-MULTIPLIER**
- **DEFAULT-CHAR-MAXLENGTH**

Si decide utilizar el parámetro **CHAR-MULTIPLIER** en DFHJS2LS, tenga en cuenta que las reglas siguientes se aplican después de que se utiliza el valor de este parámetro para calcular la cantidad de espacio necesario para los datos de caracteres.

- DFHJS2LS proporciona estas correlaciones:
 - Tipos de datos de caracteres de longitud variable que tienen una longitud máxima de más de 32 767 bytes correlacionados con un contenedor. Puede utilizar el parámetro **CHAR-VARYING-LIMIT** para establecer un límite inferior. Se crea un campo de 16 bytes en la estructura de lenguaje para almacenar el nombre del contenedor. En el tiempo de ejecución, los datos de caracteres se almacenan en un contenedor y el nombre del contenedor se coloca en la estructura de lenguaje.
 - Los tipos de datos de caracteres de longitud variable que tienen una longitud máxima de menos de 32 768 bytes se correlacionan con una estructura VARYING para todos los lenguajes excepto C/C++ y Enterprise PL/I. En C/C++, estos tipos de datos se correlacionan con series terminadas en nulo, y en Enterprise PL/I estos tipos de datos se correlacionan con estructuras VARYINGZ. Puede utilizar el parámetro **CHAR-VARYING** para seleccionar la forma en la que se correlacionan los datos de caracteres de longitud variable.
 - Los datos binarios de longitud variable que tienen una longitud máxima de menos de 32 768 bytes se correlacionan con una estructura VARYING para todos los lenguajes. Si la longitud máxima es igual o mayor de 32 768 bytes, los datos se correlacionan con un contenedor. Se crea un campo de 16 bytes en la estructura de lenguaje para almacenar el nombre del contenedor. En el tiempo de ejecución, los datos binarios se almacenan en un contenedor y el nombre del contenedor se coloca en la estructura de lenguaje.

Si tiene tipos de datos de caracteres en el esquema JSON que no tienen una longitud asociada con ellos, puede asignar una longitud predeterminada utilizando el parámetro **DEFAULT-CHAR-MAXLENGTH** en DFHJS2LS.

DFHLS2JS proporciona estas correlaciones:

- Los campos de caracteres se correlacionan con un tipo de datos string y se pueden procesar como campos de longitud fija o series terminadas en nulo en tiempo de ejecución. Puede utilizar el parámetro **CHAR-VARYING** para seleccionar la forma en la que los datos de caracteres de longitud variable se manejan en tiempo de ejecución para todos los lenguajes excepto PL/I.
- Los tipos de datos Base64Binary se correlacionan con un contenedor si la longitud máxima de los datos es mayor de 32 767 bytes o cuando la longitud no está definida. Si la longitud de los datos es 32 767 o menos, el tipo de datos base64Binary se correlaciona con una estructura VARYING para todos los lenguajes.

Nivel de correlación 1.1 y posterior

Esta opción se retiene para la compatibilidad con servicios web SOAP. No se recomienda utilizar con JSON.

Este nivel de correlación proporciona una correlación mejorada de tipos de datos binarios y caracteres JSON, en particular cuando se correlacionan datos de longitud variable que tienen las propiedades `maxLength` `minLength` definidas con valores diferentes en el esquema JSON. Los datos se manejan de la siguiente manera:

- Los tipos de datos binarios y de caracteres que tienen una longitud fija que son mayores de 16 MB se correlacionan con un contenedor para todos los lenguajes excepto PL/I. En PL/I, los tipos de datos binarios y de caracteres de longitud fija que son mayores de 32 767 se correlacionan con un contenedor. Se crea un campo de 16 bytes en la estructura de lenguaje para almacenar el nombre del contenedor. En el tiempo de ejecución, los datos de longitud fija se almacenan en un contenedor y el nombre del contenedor se coloca en la estructura de lenguaje. Puesto que los contenedores son variables en longitud, los datos de longitud fija que se correlacionan con un contenedor no se rellenan con espacios o nulos, o se truncan, para coincidir con la longitud fija especificada en el esquema JSON. Si la longitud de los datos es significativa, puede escribir su aplicación para comprobarla o activar la validación en la región de CICS. La validación tiene un impacto significativo en el rendimiento.

Solo nivel de correlación 1.1

Esta opción se retiene para la compatibilidad con servicios web SOAP. No se recomienda utilizar con JSON.

Este nivel de correlación proporciona una correlación mejorada de tipos de datos binarios y caracteres JSON, en particular cuando se correlacionan datos de longitud variable que tienen las propiedades `maxLength` `minLength` definidas con valores diferentes en el esquema JSON. Los datos se manejan de la siguiente manera:

- Los tipos de datos binarios de longitud variable se correlacionan con un contenedor. Se crea un campo de 16 bytes en la estructura de lenguaje para almacenar el nombre del contenedor. En el tiempo de ejecución, los datos binarios se almacenan en un contenedor y el nombre del contenedor se coloca en la estructura de lenguaje.
- Tipos de datos de caracteres de longitud variable que tienen una longitud máxima de más de 32 767 bytes se correlacionan con un contenedor. Se crea un campo de 16 bytes en la estructura de lenguaje para almacenar el nombre del contenedor. En el tiempo de ejecución, los datos de caracteres se almacenan en un contenedor y el nombre del contenedor se coloca en la estructura de lenguaje.
- Los tipos de datos binarios y de caracteres que tienen una longitud fija de menos de 16 MB se correlacionan con campos de longitud fija para todos los lenguajes excepto PL/I. En PL/I, los tipos de datos binarios y de caracteres de longitud fija de 32 767 o menos se correlacionan con campos de longitud fija.
- CICS codifica y decodifica datos en formato `hexBinary` pero no en formato `base64Binary`. Los tipos de datos `Base64Binary` en el esquema JSON se correlacionan con un campo en la estructura de lenguaje. El tamaño del campo se calcula utilizando la fórmula: $4 \times (\text{ceil}(z / 3))$ donde:
 - z es la longitud del tipo de datos en el esquema JSON.
 - $\text{ceil}(x)$ es el entero más pequeño mayor que o igual a x .

Si la longitud de *z* es mayor de 24 566 bytes, la estructura de lenguaje resultante no puede compilarse. Si tiene datos `base64Binary` que son más grandes de 24 566 bytes, se recomienda que utilice un nivel de correlación de 1.2. Con el nivel de correlación 1.2, puede correlacionar los datos `base64Binary` con un contenedor en lugar de utilizar un campo en la estructura de lenguaje.

Solo nivel de correlación 1.0

Esta opción se retiene para la compatibilidad con servicios web SOAP. No se recomienda utilizar con JSON.

Tenga en cuenta las siguientes limitaciones, que se han modificado en niveles de correlación posteriores:

- DFHJS2LS correlaciona tipos de datos binarios y de caracteres en el esquema JSON con campos de longitud fija en la estructura del lenguaje. Mire este esquema JSON parcial:

```
"example":{  
  "type":"string",  
  "maxLength":33000
```

El esquema JSON parcial aparece el una estructura de lenguaje COBOL como:
15 example PIC X(33000)

- CICS codifica y decodifica datos en formato `hexBinary` pero no en formato `base64Binary`. DFHJS2LS correlaciona datos `Base64Binary` con un campo de carácter de longitud fija, cuyo contenido debe estar codificado o decodificado por el programa de aplicación.
- DFHLS2JS interpreta los campos binarios y de caracteres en la estructura de lenguaje como campos de longitud fija y correlaciona esos campos con series JSON que tienen una propiedad `maxLength`. En tiempo de ejecución, los campos de la estructura de lenguaje se rellenan con espacios o nulos si no hay datos suficientes.

Correlación de lenguaje de alto nivel y esquema JSON

Uso del asistente de CICS JSON para generar correlaciones entre estructuras de lenguaje de alto nivel y esquemas JSON. El asistente de CICS JSON también genera esquemas JSON desde estructuras de datos de lenguaje de alto nivel o viceversa.

Los programas de utilidad DFHJS2LS y DFHLS2JS se conocen colectivamente como el asistente de CICS JSON.

- DFHLS2JS correlaciona estructuras de lenguaje de alto nivel con esquemas JSON.
- DFHJS2LS correlaciona esquemas JSON con estructuras de lenguaje de alto nivel.

Las dos correlaciones no son simétricas:

- Si procesa una estructura de datos de lenguaje con DFHLS2JS y, a continuación, procesa el esquema JSON resultante con DFHJS2LS, no espere que la estructura de datos final sea la misma que el original.
- Si procesa un esquema JSON con DFHJS2LS y, a continuación, procesa la estructura de lenguaje resultante con DFHLS2JS, no espere que el esquema JSON final sea el mismo que el original.
- En algunos casos, DFHJS2LS genera estructuras de lenguaje que no soporta DFHLS2JS.

Debe codificar estructuras de lenguaje de alto nivel procesadas por DFHLS2JS según las reglas del lenguaje, como se implementa en los compiladores de lenguaje que soporta CICS.

Correlación de COBOL con esquemas JSON:

El programa de utilidad DFHLS2JS soporta correlaciones entre estructuras de datos COBOL y definiciones de esquema JSON.

Los nombres COBOL se convierten en nombres JSON según las reglas siguientes:

- Los nombres duplicados se convierten en exclusivos añadiendo uno o varios dígitos numéricos.

Por ejemplo, dos instancias de year pasan a ser year y year1.

- Los guiones se sustituyen por guiones bajos. Las series de guiones contiguos son cambiados por guiones bajos contiguos.

Por ejemplo, current-user--id se convierte en current_user_id.

- Los segmentos de nombres que están delimitados por guiones y que sólo contienen caracteres en mayúsculas se convierten en minúsculas.

Por ejemplo, CA-REQUEST-ID pasa a ser ca_request_id.

- Se añade un guión bajo inicial a los nombres que empiezan por un carácter numérico.

Por ejemplo, 9A-REQUEST-ID pasa a ser _9a_request_id.

CICS correlaciona elementos de descripción de datos COBOL con elementos de esquema de acuerdo con la tabla siguiente. Los elementos de descripción de datos de COBOL que no se muestran en la tabla no son compatibles con DFHLS2JS. También se aplican las siguientes restricciones:

- Los elementos de descripción de datos con números de nivel de 66 y 77 no se soportan. Los elementos de descripción de datos con un número de nivel de 88 se ignoran.
- No se soportan las siguientes cláusulas en entradas de descripción de datos:
 - REDEFINES
 - RENAMES; es el nivel 66
 - DATE FORMAT
- Se ignoran las siguientes cláusulas en los elementos de descripción de datos:
 - BLANK WHEN ZERO
 - JUSTIFIED
 - VALUE
- Se soporta la cláusula SIGN "SIGN TRAILING". SIGN LEADING de la cláusula SIGN solo se admite cuando el nivel de correlación especificado en DFHLS2JS es 1.2 o superior.
- Se soporta SEPARATE CHARACTER en un nivel de correlación de 1.2 o superior tanto para la cláusula SIGN TRAILING como para SIGN LEADING.
- No se soportan las siguientes frases en la cláusula USAGE:
 - OBJECT REFERENCE
 - POINTER
 - FUNCTION-POINTER
 - PROCEDURE-POINTER
- Se soportan las siguientes frases en la cláusula USAGE en un nivel de correlación de 1.2 o superior:

COMPUTATIONAL-1

COMPUTATIONAL-2

- Los únicos caracteres PICTURE que se admiten para los elementos de descripción de datos DISPLAY y COMPUTATIONAL-5 son 9, S y Z.
- Los caracteres PICTURE que se admiten para los elementos de descripción de datos PACKED-DECIMAL son 9, S, V y Z.
- Los únicos caracteres PICTURE que se admiten con los elementos de descripción de datos numéricos editados son 9 y Z.
- Si el parámetro MAPPING-LEVEL se establece en 1.2 o superior y el parámetro CHAR-VARYING se establece en NULL, las matrices de caracteres se correlacionan con una string y se procesan como series terminadas en nulo.
- Si el parámetro MAPPING-LEVEL se establece en 1.2 o superior y el parámetro CHAR-VARYING se establece en BINARY, las matrices de caracteres se correlacionan con una string y se procesan como datos binarios.
- Si el parámetro MAPPING-LEVEL se establece en 1.2 o superior y el parámetro CHAR-VARYING se establece en COLLAPSE, se ignora el espacio en blanco final en las series.
- La cláusula OCCURS DEPENDING ON es compatible a nivel de correlación 4.0 o superior. Las cláusulas OCCURS DEPENDING ON complejas no son compatibles. Esto significa que OCCURS DEPENDING ON sólo se soporta para el último campo de una estructura.
- La cláusula OCCURS INDEXED BY es compatible a nivel de correlación.

Descripción de datos COBOL	Definición de esquema JSON
PIC X(<i>n</i>)	"type": "string", "maxLength": <i>n</i>
PIC A(<i>n</i>)	
PIC G(<i>n</i>) DISPLAY-1	
PIC N(<i>n</i>)	

Descripción de datos COBOL	Definición de esquema JSON
PIC S9 DISPLAY PIC S99 DISPLAY PIC S999 DISPLAY PIC S9999 DISPLAY PIC S9(<i>n</i>) DISPLAY PIC S9(<i>n</i>) COMP PIC S9(<i>n</i>) COMP-4 PIC S9(<i>n</i>) COMP-5 PIC S9(<i>n</i>) BINARY	<pre>"type":"integer", "minimum":- (<i>n</i> + 1), "maximum": <i>n</i></pre> <p>donde <i>n</i> es el valor máximo que se puede representar mediante el patrón de '9' caracteres.</p>
PIC 9 DISPLAY PIC 99 DISPLAY PIC 999 DISPLAY PIC 9999 DISPLAY PIC 9(<i>n</i>) DISPLAY PIC 9(<i>n</i>) COMP PIC 9(<i>n</i>) COMP-4 PIC 9(<i>n</i>) COMP-5 PIC 9(<i>n</i>) BINARY	<pre>"type":"integer", "minimum":0, "maximum": <i>n</i></pre> <p>donde <i>n</i> es el valor máximo que puede ser representado por el patrón de '9' caracteres.</p>

Descripción de datos COBOL	Definición de esquema JSON
PIC S9(<i>m</i>)V9(<i>n</i>) COMP-3	<pre>"type": "number", "format": "decimal", "minimum": x , "maximum": y , "multipleOf": z</pre> <p>donde:</p> <p><i>x</i> es el valor mínimo que se puede representar mediante el patrón de '9' caracteres</p> <p><i>y</i> es el valor máximo que se puede representar mediante el patrón de '9' caracteres</p> <p><i>z</i> es la unidad más pequeña disponible = 1 / 10 "</p>
PIC 9(<i>m</i>)V9(<i>n</i>) COMP-3	<pre>"type": "number", "format": "decimal", "minimum": 0, "maximum": y , "multipleOf": z</pre> <p>donde:</p> <p><i>y</i> es el valor máximo que se puede representar mediante el patrón de '9' caracteres</p> <p><i>z</i> es la unidad más pequeña disponible = 1 / 10 "</p>
PIC S9(15) COMP-3 Admitido en el nivel de correlación 3.0 cuando DATETIME=PACKED15	<pre>"type": "string", "format": "date-time"</pre> <p>El formato de la indicación de fecha y hora se define por RFC3339.</p>
PIC S9(<i>m</i>)V9(<i>n</i>) DISPLAY Se soporta en el nivel de correlación 1.2 y superior	<pre>"type": "number", "format": "decimal", "minimum": x , "maximum": y , "multipleOf": z</pre> <p>donde:</p> <p><i>x</i> es el valor mínimo que se puede representar mediante el patrón de '9' caracteres</p> <p><i>y</i> es el valor máximo que se puede representar mediante el patrón de '9' caracteres</p> <p><i>z</i> es la unidad más pequeña disponible = 1 / 10 "</p>

Descripción de datos COBOL	Definición de esquema JSON
<p>COMP-1</p> <p>Se soporta en el nivel de correlación 1.2 y superior</p> <p>Nota: La representación de datos de coma flotante hexadecimal (HFP) de IBM no es exactamente lo mismo que la representación IEEE-754-1985 utilizada para JSON. Algunos valores podrían no convertirse de forma exacta de una representación a la otra.</p> <p>Algunos valores sumamente grandes o pequeños podrían no ser válidos para los tipos de datos flotantes. Algunos valores podrían perder precisión cuando se convierten a o desde la representación HFP. Si las conversiones precisas son importantes, considere sustituir el uso de tipos de datos COMP-1 por alternativas de precisión fija.</p>	<pre>"type": "number", "format": "float"</pre>
<p>COMP-2</p> <p>Se soporta en el nivel de correlación 1.2 y superior</p> <p>Nota: La representación de datos de coma flotante hexadecimal (HFP) de IBM no es exactamente lo mismo que la representación IEEE-754-1985 utilizada para JSON. Algunos valores podrían no convertirse de forma exacta de una representación a la otra.</p> <p>Algunos valores sumamente grandes o pequeños podrían no ser válidos para los tipos de datos dobles. Algunos valores podrían perder precisión cuando se convierten a o desde la representación HFP. Si las conversiones precisas son importantes, considere sustituir el uso de tipos de datos COMP-2 por alternativas de precisión fija. .</p>	<pre>"type": "number", "format": "double"</pre>

Descripción de datos COBOL	Definición de esquema JSON
<i>descripción de datos</i> OCCURS <i>n</i> TIMES	<p>En el nivel de correlación 4.0 e inferior</p> <p>Para primitivos:</p> <pre> "type": "array" "maxItems": <i>n</i> "minItems": <i>n</i> "items": { "type": "object", "properties": { <i>name</i> : { <i>data description JSON</i> } } "required": [<i>name</i>] }</pre> <p>Para estructuras:</p> <pre> "type": "array" "maxItems": <i>n</i> "minItems": <i>n</i> "items": { <i>data description JSON</i> }</pre> <p>Donde <i>JSON de descripción de datos</i> es la representación del esquema JSON de la <i>descripción de datos</i> COBOL y <i>name</i> es el nombre de la <i>descripción de datos</i> COBOL.</p> <p>En el nivel de correlación 4.1 y superior:</p> <p>TRUNCATE-NULL-ARRAYS = DISABLED</p> <p>Ambas correlaciones estructurada y primitiva se correlacionan de la siguiente manera.</p> <pre> "type": "array" "maxItems": <i>n</i> "minItems": <i>n</i> "items": { <i>data description JSON</i> }</pre> <p>TRUNCATE-NULL-ARRAYS = ENABLED</p> <p>Las correlaciones primitivas se correlacionan como se indica arriba y las correlaciones estructuras se correlacionan de la siguiente manera.</p> <pre> "type": "array" "maxItems": <i>n</i> "minItems": 0 "items": { <i>data description JSON</i> }</pre>

Descripción de datos COBOL	Definición de esquema JSON
<i>descripción de</i> <i>datos</i> OCCURS <i>n</i> TO <i>m</i> TIMES DEPENDING ON <i>t</i> Se admite en el nivel de correlación 4.0	<pre>"field-name" : { "type": "array" , "maxItems": <i>m</i> "minItems": <i>n</i> "items": { . . . } }</pre> El contenido del elemento de matriz depende del tipo de datos utilizado.
PIC X OCCURS <i>n</i> TIMES PIC A OCCURS <i>n</i> TIMES PIC G DISPLAY-1 OCCURS <i>n</i> TIMES PIC N OCCURS <i>n</i> TIMES	<p>Cuando CHAR-OCCURS =STRING :</p> <pre>"field-name" : { "type": "string" , "maxLength": <i>n</i> }</pre> Esta es una serie.

Descripción de datos COBOL	Definición de esquema JSON
	<p>Cuando CHAR-OCCURS =ARRAY :</p> <p>En el nivel de correlación 4.0 e inferior</p> <pre> "field-name" :{ "maxItems": m , "minItems": n , "items":{ "type": "object" , "properties":{ "field-name":{ "type": "string" , "maxLength":1 } }, "required":["field-name"] } } </pre> <p>Es una matriz de caracteres individuales.</p> <p>En el nivel de correlación 4.1 y superior:</p> <p>TRUNCATE-NULL-ARRAYS = DISABLED</p> <p>Ambas correlaciones estructurada y primitiva se correlacionan de la siguiente manera.</p> <pre> "type":"array" "maxItems": n "minItems": n "items":{ data description JSON } </pre> <p>TRUNCATE-NULL-ARRAYS = ENABLED</p> <p>Las correlaciones primitivas se correlacionan como se indica arriba y las correlaciones estructuras se correlacionan de la siguiente manera.</p> <pre> "type":"array" "maxItems": n "minItems": 0 "items":{ data description JSON } </pre>

Descripción de datos COBOL	Definición de esquema JSON
PIC X OCCURS <i>n</i> TO <i>m</i> TIMES DEPENDING ON <i>t</i> PIC A OCCURS <i>n</i> TO <i>m</i> TIMES DEPENDING ON <i>t</i> PIC G DISPLAY-1 OCCURS <i>n</i> TO <i>m</i> TIMES DEPENDING ON <i>t</i> PIC N OCCURS <i>n</i> TO <i>m</i> TIMES DEPENDING ON <i>t</i>	Cuando CHAR-OCCURS =STRING : "field-name" : { "type": "string" , "maxLength": <i>m</i> "minLength": <i>n</i> }
PIC N(<i>n</i>) USAGE NATIONAL Cuando CHAR-USAGE =NATIONAL : PIC N(<i>n</i>)	En el nivel de correlación 4.0 y superior: "type": "string", "maxLength": <i>n</i> En tiempo de ejecución, CICS rellena el campo de estructura de datos de aplicación con datos UTF-16.

Correlación de esquema JSON con COBOL:

El programa de utilidad DFHJS2LS soporta correlaciones entre esquemas JSON y estructuras de datos COBOL.

Los asistentes de CICS generan nombres de campo exclusivos y válidos para variables COBOL de los nombres de elemento de esquema que utilizan las reglas siguientes:

1. Las palabras reservadas de COBOL llevan el prefijo ' X ' inicial.
Por ejemplo, DISPLAY pasa a ser XDISPLAY.
2. Caracteres que no sean A-Z, a-z, 0-9 o guión se sustituyen por ' X ' .
Por ejemplo, monthly_total pasa a ser monthlyXtotal.
3. Si el último carácter es un guión se sustituye por ' X ' .
Por ejemplo, ca-request- pasa a ser ca-requestX.

- Los nombres duplicados en el mismo ámbito se convierten en exclusivos añadiendo uno o dos dígitos numéricos a la segunda y siguientes instancias del nombre.

Por ejemplo, tres instancias de year se convierten en year , year1 y year2.

- Un esquema JSON especifica que una variable tiene una cardinalidad variable si tiene un valor "type" de "array" , y las palabras claves "minItems" y "maxItems" se omiten o tienen diferentes valores. Si el esquema especifica que la variable tiene una cardinalidad variable, los nombres de campo se crean con los sufijos "_cont" y "_num".

Para obtener más información, consulte el apartado “Matrices de variables de elementos en DFHJS2LS” en la página 283.

- Un esquema JSON especifica que una variable es opcional si no aparece en la matriz de palabra clave "required" que está asociada al tipo "object" del esquema JSON delimitado. Para los campos opcionales, se genera un campo adicional con un sufijo _num añadido al nombre de elemento. En el tiempo de ejecución, es cero para indicar que falta el valor de los datos JSON, y no cero si el valor está presente en los datos JSON.
- Los nombres de campo están limitados a 28 caracteres. Si un nombre generado, incluido cualquier prefijo o sufijo, sobrepasa esta longitud, el nombre de elemento se trunca.

DFHJS2LS correlaciona tipos de esquema con elementos de descripción de datos COBOL utilizando el nivel de correlación especificado según la tabla siguiente. Tenga en cuenta los puntos siguientes:

- Si el parámetro **MAPPING-LEVEL** se establece en 1.2 o superior y el parámetro **CHAR-VARYING** se establece en NULL, los datos de caracteres de longitud variable se correlacionan con series terminadas en nulo y se asigna un carácter adicional al terminador en nulo.
- Si el parámetro **MAPPING-LEVEL** se establece en 1.2 o superior y el parámetro **CHAR-VARYING** se establece en YES, los datos de carácter de longitud variable se correlacionan con dos elementos relacionados: un campo de longitud y un campo de datos. Por ejemplo:

```
"textString": {
  "type": "string",
  "maxLength": 10000,
  "minLength": 1
}
```

se correlaciona con:

```
15 textString-length PIC S9999 COMP-5 SYNC
15 textString PIC X(10000)
```

Palabra clave de esquema JSON	Descripción de datos COBOL
Todo de: "type": "null" "type": [] "enum": [] "allOf" "anyOf" "noneOf" "not"	No soportada

Palabra clave de esquema JSON	Descripción de datos COBOL
"\$schema": "http://json-schema.org/draft-04/schema#"	Esta palabra clave se ignora, pero se supone que es compatible con la especificación de esquema borrador 04 JSON.
"title": "same text" "description": "more text"	No se tienen en cuenta estas palabras clave.
"format": "<predefined values>"	La palabra clave "format" se utiliza para modificar la estructura generada o el valor de tiempo de ejecución. Consulte la siguiente información de la tabla para ver el uso soportado de "format".
"type": "array", "items": {<JSON Sub-schema>}, "additionalItems": false, "maxItems": m , "minItems": n	<p>La única forma de matriz JSON soportada actualmente es un número repetido de valores del mismo tipo. El <JSON Sub-schema> debe definir un "type" soportado , pero ese "type" no puede ser "array". Se trata de una restricción sobre la estructura de lenguaje generada.</p> <p>Se supone que los "additionalItems" son falsos y no se soporta ningún otro valor.</p> <p>Si "minItems" y "maxItems" están presentes, pero no iguales, la matriz se trata como una cardinalidad fija, de lo contrario, se trata como una cardinalidad variable. Consulte “Matrices de variables de elementos en DFHJS2LS” en la página 283.</p>
"type": "array", "uniqueItems": true	"uniqueItems" no se soportan con matrices JSON.

Palabra clave de esquema JSON	Descripción de datos COBOL
<pre>"type": "object", "additionalProperties": false, "properties": { ["<element name>": {<JSON Sub-schema> } [,]]* } "required": [["<element name>" [,]]*]</pre>	<p>La única forma de objeto JSON soportado actualmente es un conjunto fijo de elementos con nombre.</p> <p>Esto generará una estructura (o subestructura) utilizando los nombres de elemento.</p> <p>Se supone que las "additionalProperties" son false, y no se soporta ningún otro valor.</p> <p>Cualquier elemento del objeto "properties" se considera "optional" si no está en la matriz "required" o si no existe la matriz "required". A un elemento "optional" se le proporciona una ordinalidad variable de cero para X; donde X es 1 o el número máximo de elementos de la matriz, donde el elemento se define como una matriz. Consulte "Matrices de variables de elementos en DFHJS2LS" en la página 283.</p>
<pre>"type": "object", "maxProperties": m, "minProperties": n, "patternProperties": {}, "dependencies":</pre>	<p>No se soporta ninguna de estas palabras claves con objetos JSON.</p>
<pre>"type": "string" "maxLength": m "pattern": "regular expression", "minLength": l</pre>	<p>PIC X(z)</p> <p>donde el valor de z se basa en m , pero depende de los valores del parámetro CHAR-VARYING.</p> <p>m se basa en la palabra clave "maxLength" y se trata como una serie de longitud fija.</p> <p>Las restricciones "pattern" y "minLength" se pasan a través de la estructura de lenguaje sólo como un comentario.</p>
<pre>"type": "string" "maxLength": m</pre>	<p>Cuando CCSID=1200 está en el nivel de correlación 4.0 y superior:</p> <p>PIC N(z) USAGE NATIONAL</p> <p>donde el valor de z se basa en m , pero depende de los valores del parámetro CHAR-VARYING.</p> <p>m se basa en la palabra clave "maxLength" y se trata como una serie de longitud fija.</p>

Palabra clave de esquema JSON	Descripción de datos COBOL
<pre>"*name*":{ "type":"string", "format":"date-time" }</pre>	<p>PIC S9(15) COMP-3</p> <p>Todo soportado cuando DATETIME=PACKED15</p> <p>Tenga en cuenta que "maxLength" y "minLength" no se soportan para este formato</p>
<pre>"*name*":{ "type":"string", "format":"uri" }</pre>	<p>PIC X(<i>m</i>)</p> <p>donde <i>m</i> se basa en la palabra clave "maxLength" y se trata como una serie de longitud fija.</p> <p>Cuando CCSID=1200 está en el nivel de correlación 4.0 y superior:</p> <p>PIC N(<i>m</i>) USAGE NATIONAL</p> <p>donde <i>m</i> se basa en la palabra clave "maxLength" y se trata como una serie de longitud fija.</p>
<pre>"*name*":{ "type":"string", "format":"base64Binary" }</pre>	<p>PIC X(<i>m</i>)</p> <p>donde <i>m</i> se basa en la palabra clave "maxLength"</p>
<pre>"*name*":{ "type":"string", "format":"hexBinary" }</pre>	<p>PIC X(<i>m</i>)</p> <p>donde <i>m</i> se basa en la palabra clave "maxLength"</p>
<pre>"*name*":{ "type":"string", "format":"<predefined>" }</pre>	<p>PIC X(<i>m</i>)</p> <p>donde <i>m</i> se basa en la palabra clave "maxLength" y se trata como una serie de longitud fija, y donde <predefined> es uno de: <i>email</i> , <i>hostname</i> , <i>ipv4</i> o <i>ipv6</i> . Se utiliza un "pattern" relevante y se pasa al comentario.</p> <p>Cuando CCSID=1200 está en el nivel de correlación 4.0 y superior:</p> <p>PIC N(<i>m</i>) USAGE NATIONAL</p> <p>donde <i>m</i> se basa en la palabra clave "maxLength" y se trata como una serie de longitud fija, y donde <predefined> es uno de: <i>email</i> , <i>hostname</i> , <i>ipv4</i> o <i>ipv6</i> . Se utiliza un "pattern" relevante y se pasa al comentario.</p>
<pre>"type":"boolean"</pre>	<p>PIC X DISPLAY</p> <p>El valor x'00' implica falso, x'01' implica verdadero.</p>

Palabra clave de esquema JSON	Descripción de datos COBOL
"type": "integer", "maxExclusive": true, "minExclusive": true, "multipleOf": n	Las restricciones "maxExclusive" y "minExclusive" se pasan a la estructura de lenguaje sólo como un comentario. "multipleOf" se ignora.
"type": "integer", minimum=0, maximum=255	Nivel de correlación 3.0 e inferior: PIC X DISPLAY Nivel de correlación 4.0 y superior (o cuando se ha especificado el parámetro INTEGER-AS-PIC9, independientemente del nivel de correlación) : PIC 9(z) COMP-5 SYNC o PIC 9(z) DISPLAY donde $10^{(z-1)} < m \leq 10^z$
"type": "integer", minimum:-128, maximum:127	Nivel de correlación 3.0 e inferior: PIC X DISPLAY Nivel de correlación 4.0 y superior (o cuando se ha especificado el parámetro INTEGER-AS-PIC9, independientemente del nivel de correlación) : PIC S9(z) COMP-5 SYNC o PIC S9(z) DISPLAY donde $10^{(z-1)} < m \leq 10^z$
"type": "integer", minimum:0, maximum; m	PIC 9(z) COMP-5 SYNC o PIC 9(z) DISPLAY donde $10^{(z-1)} < m \leq 10^z$
"type": "integer", minimum:- m , maximum: m -1	PIC S9(z) COMP-5 SYNC o PIC S9(z) DISPLAY donde $10^{(z-1)} < m \leq 10^z$

Palabra clave de esquema JSON	Descripción de datos COBOL
<pre>"type": "number", "maximum": m, "minimum": n, "maxExclusive": true, "minExclusive": true, "multipleOf": n</pre>	<p>"maximum", "minimum", "maxExclusive" y "minExclusive" se pasan a la estructura de lenguaje sólo como un comentario.</p> <p>"multipleOf" se ignora.</p>
<pre>"type": "number" "format": "decimal"</pre>	<p>PIC 9(<i>p</i>)V9(<i>n</i>) COMP-3</p> <p>donde <i>p</i> y <i>n</i> son valores predeterminados.</p>
<pre>"type": "number" "format": "float"</pre>	<p>Nivel de correlación 1.1 e inferior:</p> <ul style="list-style-type: none"> PIC X(32) <p>Nivel de correlación 1.2 y superior:</p> <ul style="list-style-type: none"> COMP-1 <p>Nota: La representación de datos de coma flotante hexadecimal (HFP) de IBM no es exactamente lo mismo que la representación IEEE-754-1985 utilizada para JSON. Algunos valores podrían no convertirse de forma exacta de una representación a la otra.</p> <p>Algunos valores sumamente grandes o pequeños podrían no ser válidos para los tipos de datos flotantes. Algunos valores podrían perder precisión cuando se convierten a o desde la representación HFP. Si las conversiones precisas son importantes, considere sustituir el uso de tipos de datos COMP-1 por alternativas de precisión fija.</p>
<pre>"type": "number" "format": "double"</pre>	<p>Nivel de correlación 1.1 e inferior:</p> <ul style="list-style-type: none"> PIC X(32) <p>Nivel de correlación 1.2 y superior:</p> <ul style="list-style-type: none"> COMP-2 <p>Nota: La representación de datos de coma flotante hexadecimal (HFP) de IBM no es exactamente lo mismo que la representación IEEE-754-1985 utilizada para JSON. Algunos valores podrían no convertirse de forma exacta de una representación a la otra.</p> <p>Algunos valores sumamente grandes o pequeños podrían no ser válidos para los tipos de datos dobles. Algunos valores podrían perder precisión cuando se convierten a o desde la representación HFP. Si las conversiones precisas son importantes, considere sustituir el uso de tipos de datos COMP-2 por alternativas de precisión fija. .</p>

Nota: CICS no puede transformar valores enteros mayores que el valor máximo para una serie larga firmada ($2^{63} - 1$) a menos que esté encerrada entre comillas.

Nota: Los valores mínimo y máximo especificados en el esquema para tipos numéricos se utilizan únicamente para correlacionar con un tipo de datos COBOL. Los datos no se validan con estos valores durante el tiempo de ejecución.

Algunos de los tipos de esquema que se muestran en la tabla se correlacionan con un formato COBOL de COMP-5 SYNC o de DISPLAY, dependiendo de los valores (si los hubiera) que se especifican en las palabras clave minimum y maximum:

- Para tipos firmados (short , int y long), se utiliza DISPLAY cuando se especifica lo siguiente:

```
"maximum":  
a  
"minimum":  
-a
```

donde *a* es una serie de '9'.

- Para los tipos sin firmar (unsignedShort , unsignedInt y unsignedLong), se utiliza DISPLAY cuando se especifica lo siguiente:

```
"maximum":  
a  
"minimum":0
```

donde *a* es una serie de '9'.

Cuando se especifica otro valor, o no se especifica ningún valor, se utiliza COMP-5 SYNC.

Correlación de C y C++ con esquemas JSON:

El programa de utilidad DFHLS2JS soporta correlaciones entre tipos de datos C y C++ y definiciones de esquema JSON.

Los nombres C y C++ se convierten en nombres JSON según las reglas siguientes:

1. Los caracteres que no son válidos en los nombres de propiedad JSON se sustituyen por 'X'.
Por ejemplo, monthly-total pasa a ser monthlyXtotal.
2. Los nombres duplicados se convierten en exclusivos añadiendo uno o varios dígitos numéricos.
Por ejemplo, dos instancias de year pasan a ser year y year1.

DFHLS2JS correlaciona tipos de datos C y C++ con elementos de esquema según la tabla siguiente. Los tipos C y C++ que no se muestran en la tabla no son compatibles con DFHLS2JS. El calificador `_Packed` se soporta para las estructuras. Se aplican estas restricciones:

- Los archivos de cabecera deben contener una instancia de alto nivel struct.
- No puede declarar un tipo de estructura que se contiene a sí mismo como miembro.
- Los siguientes tipos de datos C y C++ no se soportan:
decimal
long double

wchar_t (sólo C++)

- Los siguientes caracteres se ignoran si están presentes en el archivo de cabecera.

Especificadores de clase de almacenamiento:

auto
register
static
extern
mutable

Calificadores

const
volatile
_Export (sólo C++)

Especificadores de función

inline (sólo C++)
Virtual (sólo C++)

Valores iniciales

- El archivo de cabecera no debe contener estos elementos:

Uniones

Declaraciones de clase

Tipos de datos de enumeración

Variables de tipo de puntero

Declaraciones de plantilla

Las macros predefinidas; es decir, macros son nombres que comienzan y finalizan con dos caracteres de subrayado (__)

La secuencia de continuación de línea (un símbolo \ que va seguido inmediatamente por un carácter de nueva línea)

Declaradores de funciones de prototipos

Directivas de preprocesamiento

Campos de bits

La palabra clave __cdecl (o _cdecl) (sólo C++)

- El programador de aplicación debe utilizar un compilador de 32 bits para asegurar que un int se correlaciona con 4 bytes.

- No se soportan las siguientes palabras clave de C++ reservadas:

explicit
using
namespace
typename
typeid

- Si el parámetro **MAPPING-LEVEL** se establece en 1.2 o superior y el parámetro **CHAR-VARYING** se establece en NULL, las matrices de caracteres se correlacionan con una string y se procesan como series terminadas en nulo.
- Si el parámetro **MAPPING-LEVEL** se establece en 1.2 o superior y el parámetro **CHAR-VARYING** se establece en BINARY, las correlaciones de caracteres se correlacionan con xsd:base64Binary y se procesan como datos binarios.
- Si el parámetro **MAPPING-LEVEL** se establece en 1.2 o superior y el parámetro **CHAR-VARYING** se establece en COLLAPSE, <xsd:whiteSpace value="collapse"/> se genera para las series.

Tipos de datos C y C++	Esquema simpleType
char[<i>z</i>]	"type":"string" "maxLength": <i>z</i>
char16_t[<i>n</i>]	En el nivel de correlación 4.0 y superior: "type":"string" "maxLength": <i>n</i> En tiempo de ejecución, CICS rellena el campo de estructura de datos de aplicación con datos UTF-16.
char[8] Se soporta en el nivel de correlación 3.0 y superior cuando DATETIME=PACKED15	"type":"string" "format":"date-time" El formato de la indicación de fecha y hora se define por RFC3339.
char short int long long long	"type":"integer", "minimum":- (<i>n</i> + 1), "maximum": <i>n</i> donde <i>n</i> es el valor máximo que puede representarse por el primitivo.
unsigned char unsigned short unsigned int unsigned long unsigned long long	"type":"integer", "minimum":0, "maximum": <i>n</i> donde <i>n</i> es el valor máximo que puede representarse por el primitivo.
bool (sólo C++)	"type":"boolean"
float Se soporta en un nivel de correlación 1.2 y superiores.	"type":"number", "format":"float" Nota: La representación de datos de coma flotante hexadecimal (HFP) de IBM no es exactamente lo mismo que la representación IEEE-754-1985 utilizada para JSON. Algunos valores podrían no convertirse de forma exacta de una representación a la otra. Algunos valores sumamente grandes o pequeños podrían no ser válidos para los tipos de datos flotantes. Algunos valores podrían perder precisión cuando se convierten a o desde la representación HFP. Si las conversiones precisas son importantes, considere sustituir el uso de tipos de datos flotantes por alternativas de precisión fija.
double Se soporta en un nivel de correlación 1.2 y superiores.	"type":"number", "format":"double" Nota: La representación de datos de coma flotante hexadecimal (HFP) de IBM no es exactamente lo mismo que la representación IEEE-754-1985 utilizada para JSON. Algunos valores podrían no convertirse de forma exacta de una representación a la otra. Algunos valores sumamente grandes o pequeños podrían no ser válidos para los tipos de datos dobles. Algunos valores podrían perder precisión cuando se convierten a o desde la representación HFP. Si las conversiones precisas son importantes, considere sustituir el uso de tipos de datos dobles por alternativas de precisión fija.

Tipos de datos C y C++	Esquema simpleType
<i>type name</i> [<i>n</i>]	<p>Para primitivos:</p> <pre> "type": "array" "maxItems": <i>n</i> "minItems": <i>n</i> "items": { "type": "object", "properties": { <i>name</i> } : { <i>type JSON</i> } } "required": [<i>name</i>] </pre> <p>Para estructuras:</p> <pre> "type": "array" "maxItems": <i>n</i> "minItems": <i>n</i> "items": { <i>type JSON</i> } </pre> <p>donde <i>type JSON</i> es la representación del esquema JSON del tipo C o C++.</p>

Correlación de esquema JSON con C y C++:

Los programas de utilidad DFHJS2LS soportan correlaciones entre esquemas JSON y tipos de datos C y C++.

Los asistentes de CICS generan nombres de campo exclusivos y válidos para variables C y C++ de los nombres de elemento de esquema que utilizan las reglas siguientes:

1. Caracteres que no sean A-Z, a-z, 0-9, o _ se sustituyen por ' X '.
Por ejemplo, `monthly-total` pasa a ser `monthlyXtotal`.
2. Si el primer carácter no es un carácter alfabético, se sustituye por una ' X ' inicial.
Por ejemplo, `_monthlysummary` pasa a ser `Xmonthlysummary`.
3. Los nombres duplicados en el mismo ámbito se convierten en exclusivos añadiendo uno o dos dígitos numéricos a la segunda y siguientes instancias del nombre.
Por ejemplo, tres instancias de `year` pasan a ser `year` , `year1` y `year2`.
4. Un esquema JSON especifica que una variable tiene una cardinalidad variable si tiene un valor "type" de "array" , y las palabras claves "minItems" y "maxItems" se omiten o tienen diferentes valores. Si el esquema especifica que la variable tiene una cardinalidad variable, los nombres de campo se crean con los sufijos "_cont" y "_num".
Para obtener más información, consulte el apartado “Matrices de variables de elementos en DFHJS2LS” en la página 283.

5. Un esquema JSON especifica que una variable es opcional si no aparece en la matriz de palabra clave "required" que está asociada al tipo "object" del esquema JSON delimitado. Para los campos opcionales, se genera un campo adicional con un sufijo _num añadido al nombre de elemento. En el tiempo de ejecución, es cero para indicar que falta el valor de los datos JSON, y no cero si el valor está presente en los datos JSON.
6. Los nombres de campo están limitados a 50 caracteres. Si un nombre generado, incluidos prefijos y sufijos, supera esta longitud, el nombre del elemento se trunca.

DFHJS2LS correlaciona valores de tipo de esquema JSON con tipos de datos C y C++ según la tabla siguiente. También se aplican las reglas siguientes:

- Si el parámetro **MAPPING-LEVEL** se establece en 1.2 o superior y el parámetro **CHAR-VARYING** se establece en NULL, los datos de caracteres de longitud variable se correlacionan con series terminadas en nulo y se asigna un carácter adicional al terminador en nulo.
- Si el parámetro **MAPPING-LEVEL** se establece en 1.2 o superior y el parámetro **CHAR-VARYING** se establece en YES, los datos de carácter de longitud variable se correlacionan con dos elementos relacionados: un campo de longitud y un campo de datos.

Palabra clave de esquema JSON	Tipos de datos C y C++
Todo de: "type": "null" "type": [] "enum": [] "allOf" "anyOf" "noneOf" "not"	No soportado
"\$schema": "http://json-schema.org/draft-04/schema#"	Esta palabra clave se ignora, pero se supone que es compatible con la especificación de esquema borrador 04 JSON.
"title": "same text" "description": "more text"	No se tienen en cuenta estas palabras clave.
"format": "<predefined values>"	La palabra clave "format" se utiliza para modificar la estructura generada o el valor de tiempo de ejecución. Consulte la siguiente información para ver el uso soportado de "format".

Palabra clave de esquema JSON	Tipos de datos C y C++
<pre>"type": "array", "items": {<JSON Sub-schema>}, "additionalItems": false, "maxItems": m , "minItems": n</pre>	<p>La única forma de matriz JSON soportada actualmente es un número repetido de valores del mismo tipo. El <JSON Sub-schema> debe definir un "type" soportado , pero ese "type" no puede ser "array". Se trata de una restricción sobre la estructura de lenguaje generada.</p> <p>Se supone que los "additionalItems" son falsos y no se soporta ningún otro valor.</p> <p>Si "minItems" y "maxItems" están presentes, pero no iguales, la matriz se trata como una cardinalidad fija, de lo contrario, se trata como una cardinalidad variable. Consulte “Matrices de variables de elementos en DFHJS2LS” en la página 283.</p>
<pre>"type": "array", "uniqueItems": true</pre>	<p>"uniqueItems" no se soportan con matrices JSON. El <JSON Sub-schema> debe definir un "type" soportado , pero ese "type" no puede ser "array" . Se trata de una restricción sobre la estructura de lenguaje generada.</p>
<pre>"type": "object", "additionalProperties": false, "properties": { ["<element name>": {<JSON Sub-schema>} [,]]* } "required": [["<element name>" [,]]*]</pre>	<p>La única forma de objeto JSON soportado actualmente es un conjunto fijo de elementos con nombre.</p> <p>Esto generará una estructura (o subestructura) utilizando los nombres de elemento.</p> <p>Se supone que las "additionalProperties" son false, y no se soporta ningún otro valor.</p> <p>Cualquier elemento del objeto "properties" se considera "optional" si no está en la matriz "required" o si no existe la matriz "required". A un elemento "optional" se le proporciona una ordinalidad variable de cero para X; donde X es 1 o el número máximo de elementos de la matriz, donde el elemento se define como una matriz. Consulte “Matrices de variables de elementos en DFHJS2LS” en la página 283.</p>
<pre>"type": "object", "maxProperties": m, "minProperties": n, "patternProperties": {}, "dependencies":</pre>	<p>No se soporta ninguna de estas palabras claves con objetos JSON.</p>

Palabra clave de esquema JSON	Tipos de datos C y C++
<pre>"type": "string" "maxLength": m "pattern": "regular expression>", "minLength": l</pre>	<p>char[z]</p> <p>donde el valor de z se basa en <i>m</i> , pero depende de los valores del parámetro CHAR-VARYING.</p> <p><i>m</i> se basa en la palabra clave "maxLength" y se trata como una serie de longitud fija.</p> <p>Las restricciones "pattern" y "minLength" se pasan a través de la estructura de lenguaje sólo como un comentario.</p>
<pre>"type": "string" "maxLength": m</pre>	<p>Cuando CCSID=1200 está en el nivel de correlación 4.0 y superior:</p> <p>char16_t[z]</p> <p>donde el valor de z se basa en <i>m</i> , pero depende de los valores del parámetro CHAR-VARYING.</p> <p><i>m</i> se basa en la palabra clave "maxLength" y se trata como una serie de longitud fija.</p>
<pre>"*name*": { "type": "string", "format": "date-time" }</pre>	<p>char[8]</p> <p>Todo soportado cuando DATETIME=PACKED15</p>
<pre>"*name*": { "type": "string", "format": "uri" }</pre>	<p>char[m]</p> <p>donde <i>m</i> se basa en la palabra clave "maxLength" y se trata como una serie de longitud fija.</p> <p>Cuando CCSID=1200 está en el nivel de correlación 4.0 y superior:</p> <p>char16_t[m]</p> <p>donde <i>m</i> se basa en la palabra clave "maxLength" y se trata como una serie de longitud fija.</p>
<pre>"*name*": { "type": "string", "format": "base64Binary" }</pre>	<p>char[m]</p> <p>donde <i>m</i> se basa en la palabra clave "maxLength".</p>
<pre>"*name*": { "type": "string", "format": "hexBinary" }</pre>	<p>char[m]</p> <p>donde <i>m</i> se basa en la palabra clave "maxLength".</p>

Palabra clave de esquema JSON	Tipos de datos C y C++
<pre>"*name*":{ "type":"string", "format":"<predefined>" }</pre>	<p>char[<i>m</i>]</p> <p>donde <i>m</i> se basa en la palabra clave "maxLength" y se trata como una serie de longitud fija, y donde <predefined> es uno de: <i>email</i> , <i>hostname</i> , <i>ipv4</i> o <i>ipv6</i>. Se utiliza un "pattern" y se pasa al comentario.</p> <p>Cuando CCSID=1200 está en el nivel de correlación 4.0 y superior:</p> <p>char16_t[<i>m</i>]</p> <p>donde <i>m</i> se basa en la palabra clave "maxLength" y se trata como una serie de longitud fija, y donde <predefined> es uno de: <i>email</i> , <i>hostname</i> , <i>ipv4</i> o <i>ipv6</i> . Se utiliza un "pattern" relevante y se pasa al comentario.</p>
"type":"boolean"	<p>bool (sólo C++)</p> <p>short (sólo C)</p>
<pre>"type": "integer", "maxExclusive": true, "minExclusive": true, "multipleOf": n</pre>	<p>Las restricciones "maxExclusive" y "minExclusive" se pasan a la estructura de lenguaje sólo como un comentario.</p> <p>"multipleOf" se ignora.</p>
<pre>"type":"integer", minimum:-128, maximum:127</pre>	signed char
<pre>"type":"integer", minimum:0, maximum:255</pre>	unsigned char
<pre>"type":"integer", minimum:-32768, maximum:32767</pre>	short
<pre>"type":"integer", minimum:0, maximum:65535</pre>	unsigned short
<pre>"type":"integer", minimum:-2147483648, maximum:2147483647</pre>	int
<pre>"type":"integer", minimum:0, maximum:4294967295</pre>	unsigned int
<pre>"type":"integer", minimum:-9223372036854775808, maximum:9223372036854775807</pre>	long long
<pre>"type":"integer", minimum:0, maximum:18446744073709551615</pre>	unsigned long long

Palabra clave de esquema JSON	Tipos de datos C y C++
<pre>"type": "number", "maximum": m, "minimum": n, "maxExclusive": true, "minExclusive": true, "multipleOf": n</pre>	<p>Las restricciones "maximum", "minimum", "maxExclusive" y "minExclusive" se pasan a la estructura de lenguaje sólo como un comentario.</p> <p>"multipleOf" se ignora.</p>
<pre>"type": "number" "format": "float"</pre>	<p>Nivel de correlación 1.1 e inferior:</p> <ul style="list-style-type: none"> char[32] <p>Nivel de correlación 1.2 y superior:</p> <ul style="list-style-type: none"> float(*) <p>Nota: La representación de datos de coma flotante hexadecimal (HFP) de IBM no es exactamente lo mismo que la representación IEEE-754-1985 utilizada para JSON. Algunos valores podrían no convertirse de forma exacta de una representación a la otra. Algunos valores sumamente grandes o pequeños podrían no ser válidos para los tipos de datos flotantes. Algunos valores podrían perder precisión cuando se convierten a o desde la representación HFP. Si las conversiones precisas son importantes, considere sustituir el uso de tipos de datos flotantes por alternativas de precisión fija.</p>
<pre>"type": "number" "format": "double"</pre>	<p>Nivel de correlación 1.0 e inferior:</p> <ul style="list-style-type: none"> char[32] <p>Nivel de correlación 1.2 y superior:</p> <ul style="list-style-type: none"> double(*) <p>Nota: La representación de datos de coma flotante hexadecimal (HFP) de IBM no es exactamente lo mismo que la representación IEEE-754-1985 utilizada para JSON. Algunos valores podrían no convertirse de forma exacta de una representación a la otra. Algunos valores sumamente grandes o pequeños podrían no ser válidos para los tipos de datos dobles. Algunos valores podrían perder precisión cuando se convierten a o desde la representación HFP. Si las conversiones precisas son importantes, considere sustituir el uso de tipos de datos dobles por alternativas de precisión fija.</p>

Nota: CICS no puede transformar valores enteros mayores que el valor máximo para una serie larga firmada ($2^{63} - 1$) a menos que esté encerrada entre comillas.

Nota: Los valores mínimo y máximo especificados en el esquema para tipos numéricos se utilizan únicamente para correlacionar con un tipo de datos C o C++. Los datos no se validan con estos valores durante el tiempo de ejecución.

Correlación de PL/I con esquemas JSON:

Los programas de utilidad DFHLS2JS y DFHLS2WS soportan correlaciones entre estructuras de datos de PL/I y definiciones de esquema JSON. Puesto que el compilador de Enterprise PL/I y los compiladores de PL/I más antiguos difieren, se admiten dos opciones de lenguaje: PLI-ENTERPRISE y PLI-OTHER.

Los nombres PL/I se convierten en nombres JSON según las reglas siguientes:

1. Los caracteres que no son válidos en los nombres de propiedad JSON se sustituyen por 'x'.
Por ejemplo, `monthly$total` pasa a ser `monthlyxtotal`.
2. Los nombres duplicados se convierten en exclusivos añadiendo uno o varios dígitos numéricos.

Por ejemplo, dos instancias de `year` pasan a ser `year` y `year1`.

DFHLS2JS correlaciona tipos de datos PL/I con elementos de esquema según la tabla siguiente. Los tipos PL/I que no se muestran en la tabla no son compatibles con DFHLS2JS. También se aplican las siguientes restricciones:

- Los elementos de datos con el atributo **COMPLEX** no se soportan.
- Los elementos de datos con atributo **FLOAT** se soportan en un nivel de correlación de 1.2 o superior. Enterprise PL/I **FLOAT IEEE** no se soporta.
- Las series DBCS puras **VARYING** y **VARYINGZ** se soportan en un nivel de correlación de 1.2 o superior.
- Los elementos de datos que se especifican con **DECIMAL**(*p* , *q*) se soportan únicamente cuando $p \geq q$.
- Los elementos de datos que se especifican con **BINARY**(*p* , *q*) se soportan únicamente cuando $q = 0$.
- Si se especifica el atributo **PRECISION** para un elemento de datos, se ignora.
- No se soportan las series **PICTURE**.
- Los elementos de datos **ORDINAL** se tratan como tipos de datos **FIXED BINARY(7)**.
- Si el parámetro **MAPPING-LEVEL** se establece en 1.2 o superior y el parámetro **CHAR-VARYING** se establece en **NULL**, las matrices de caracteres se correlacionan con una `string` y se procesan como series terminadas en nulo; esta correlación no se aplica a Enterprise PL/I.
- Si el parámetro **MAPPING-LEVEL** se establece en 1.2 o superior y el parámetro **CHAR-VARYING** se establece en **BINARY**, las correlaciones de caracteres se correlacionan con `string` y se procesan como datos binarios.
- Si el parámetro **MAPPING-LEVEL** se establece en 1.2 o superior y el parámetro **CHAR-VARYING** se establece en **COLLAPSE**; los espacios en blanco inicial y final se eliminarán y varios espacios se sustituyen por un sólo espacio.

DFHLS2JS no implementa completamente algoritmos de relleno de PL/I; por lo tanto, debe declarar bytes de relleno explícitamente en la estructura de datos. DFHLS2JS emite un mensaje si detecta que faltan los bytes de relleno. Cada estructura de nivel superior debe comenzar con un límite de palabra doble y cada byte de la estructura debe correlacionarse con el límite correcto. Tenga presente este fragmento de código:

```
3 FIELD1 FIXED BINARY(7),  
3 FIELD2 FIXED BINARY(31),  
3 FIELD3 FIXED BINARY(63);
```

En este ejemplo:

- FIELD1 tiene 1 byte de longitud y se puede alinear con cualquier límite.
- FIELD2 tiene 4 bytes de longitud y debe alinearse con un límite de palabra completa.
- FIELD3 tiene 8 bytes de longitud y debe alinearse con un límite de palabra doble.

El compilador Enterprise PL/I alinea los campos en el orden siguiente:

1. FIELD3 se alinea primero porque tiene los requisitos de límite más fuertes.
2. FIELD2 se alinea en un límite de palabra completa inmediatamente antes de FIELD3.
3. FIELD1 se alinea en un límite de byte inmediatamente antes de FIELD3.

Por último, para que toda la estructura se alinee en un límite de palabra completa, el compilador inserta tres bytes de relleno inmediatamente antes de FIELD1.

Puesto que DFHLS2JS no inserta bytes de relleno equivalentes, debe declararlos explícitamente antes de que DFHLS2JS procese la estructura. Por ejemplo:

```
3 PAD1 FIXED BINARY(7),  
3 PAD2 FIXED BINARY(7),  
3 PAD3 FIXED BINARY(7),  
3 FIELD1 FIXED BINARY(7),  
3 FIELD2 FIXED BINARY(31),  
3 FIELD3 FIXED BINARY(63);
```

De forma alternativa, puede cambiar la estructura para declarar todos los campos como no alineados y volver a compilar la aplicación que utiliza la estructura. Para obtener más información sobre los requisitos de alineación de memoria estructural de PL/I, consulte *Enterprise PL/I Language Reference*.

Descripción de datos PL/I	Definición de esquema JSON
FIXED BINARY (<i>n</i>)	<pre>"type":"integer", "minimum":- (<i>n</i> + 1), "maximum": <i>n</i></pre> <p>donde <i>n</i> es el valor máximo que puede representarse por el primitivo.</p>
UNSIGNED FIXED BINARY(<i>n</i>) Restricción: Sólo Enterprise PL/I	<pre>"type":"integer", "minimum":0, "maximum": <i>n</i></pre> <p>donde <i>n</i> es el valor máximo que puede representarse por el primitivo.</p>

Descripción de datos PL/I	Definición de esquema JSON
FIXED DECIMAL(<i>n</i> , <i>m</i>)	<pre>"type":"number", "format":"decimal", "multipleOf": x , "maximum": y , "minimum":- z</pre> <p>donde:</p> <p><i>x</i> es la unidad más pequeña disponible = $1 / 10^m$</p> <p><i>y</i> es el valor máximo que se puede representar mediante la combinación de <i>n</i> y <i>m</i></p> <p><i>z</i> es el valor máximo que se puede representar mediante la combinación de <i>n</i> y <i>m</i></p>
FIXED DECIMAL(15) Se soporta en el nivel de correlación 3.0 y superior cuando DATETIME=PACKED15	<pre>"type":"string", "format":"date-time"</pre> <p>El formato de la indicación de fecha y hora se define por RFC3339.</p>
BIT(<i>n</i>) donde <i>n</i> es un múltiplo de 8. No se soportan otros valores.	<pre>"type":"string" "maxLength": m</pre> <p>donde $m = n / 8$</p>
CHARACTER(<i>n</i>) VARYING y VARYINGZ también se soportan en el nivel de correlación 1.2 y superior. Restricción: VARYINGZ sólo se soporta en Enterprise PL/I	<pre>"type":"string" "maxLength": n</pre>
GRAPHIC(<i>n</i>) VARYING y VARYINGZ también se soportan en el nivel de correlación 1.2 y superior. Restricción: VARYINGZ sólo es compatible con Enterprise PL/I	<p>En un nivel de correlación de 1.0 y 1.1, donde $m = 2 * n$:</p> <pre>"type":"string" "maxLength": m</pre> <p>En un nivel de correlación 1.2 o superior:</p> <pre>"type":"string" "maxLength": n</pre>

Descripción de datos PL/I	Definición de esquema JSON
<p>WIDECHAR(<i>n</i>)</p> <p>Restricción: Sólo Enterprise PL/I</p>	<p>En un nivel de correlación de 1.0 y 1.1, donde $m = 2 * n$:</p> <pre>"type": "string" "maxLength": m</pre> <p>En un nivel de correlación 1.2 o superior:</p> <pre>"type": "string" "maxLength": n</pre> <p>En un nivel de correlación 4.0 y superior, CICS llena el campos de estructura de datos de aplicación con datos UTF-16.</p> <pre><xsd:simpleType> <xsd:restriction base="xsd:string"> <xsd:maxLength value="n"/> <xsd:whiteSpace value="preserve"/> </xsd:restriction> </xsd:simpleType></pre>
<p>ORDINAL</p> <p>Restricción: Sólo Enterprise PL/I</p>	<pre>"type": "integer", "minimum": 0, "maximum": 255</pre>
<p>BINARY FLOAT(<i>n</i>)</p> <p>donde $n \leq 21$</p> <p>Se soporta en un nivel de correlación 1.2 y superiores.</p> <p>Nota: La representación de datos de coma flotante hexadecimal (HFP) de IBM no es exactamente lo mismo que la representación IEEE-754-1985 utilizada para JSON. Algunos valores podrían no convertirse de forma exacta de una representación a la otra. Algunos valores sumamente grandes o pequeños podrían no ser válidos para los tipos de datos flotantes. Algunos valores podrían perder precisión cuando se convierten a o desde la representación HFP. Si las conversiones precisas son importantes, considere sustituir el uso de tipos de datos BINARY FLOAT por alternativas de precisión fija.</p>	<pre>"type": "number", "format": "float"</pre>

Descripción de datos PL/I	Definición de esquema JSON
<p>BINARY FLOAT(n)</p> <p>donde $21 < n \leq 53$</p> <p>No se soportan los valores superiores a 53.</p> <p>Se soporta en un nivel de correlación 1.2 y superiores.</p> <p>Nota: La representación de datos de coma flotante hexadecimal (HFP) de IBM no es exactamente lo mismo que la representación IEEE-754-1985 utilizada para JSON. Algunos valores podrían no convertirse de forma exacta de una representación a la otra. Algunos valores sumamente grandes o pequeños podrían no ser válidos para los tipos de datos flotantes. Algunos valores podrían perder precisión cuando se convierten a o desde la representación HFP. Si las conversiones precisas son importantes, considere sustituir el uso de tipos de datos BINARY FLOAT por alternativas de precisión fija.</p>	<pre>"type": "number", "format": "double"</pre>
<p>DECIMAL FLOAT(n)</p> <p>donde $n \leq 6$</p> <p>Se soporta en un nivel de correlación 1.2 y superiores.</p> <p>Nota: La representación de datos de coma flotante hexadecimal (HFP) de IBM no es exactamente lo mismo que la representación IEEE-754-1985 utilizada para JSON. Algunos valores podrían no convertirse de forma exacta de una representación a la otra. Algunos valores sumamente grandes o pequeños podrían no ser válidos para los tipos de datos flotantes. Algunos valores podrían perder precisión cuando se convierten a o desde la representación HFP. Si las conversiones precisas son importantes, considere sustituir el uso de tipos de datos DECIMAL FLOAT por alternativas de precisión fija.</p>	<pre>"type": "number", "format": "float"</pre>

Descripción de datos PL/I	Definición de esquema JSON
<p>DECIMAL FLOAT(<i>n</i>)</p> <p>donde $6 < n \leq 16$</p> <p>No se soportan los valores superiores a 16.</p> <p>Se soporta en un nivel de correlación 1.2 y superiores.</p> <p>Nota: La representación de datos de coma flotante hexadecimal (HFP) de IBM no es exactamente lo mismo que la representación IEEE-754-1985 utilizada para JSON. Algunos valores podrían no convertirse de forma exacta de una representación a la otra. Algunos valores sumamente grandes o pequeños podrían no ser válidos para los tipos de datos flotantes. Algunos valores podrían perder precisión cuando se convierten a o desde la representación HFP. Si las conversiones precisas son importantes, considere sustituir el uso de tipos de datos DECIMAL FLOAT por alternativas de precisión fija.</p>	<pre>"type": "number", "format": "double"</pre>
<p><i>name (n) descripción de datos</i></p>	<p>Para primitivos:</p> <pre>"type": "array" "maxItems": n "minItems": n "items": { "type": "object", "properties": { name : { data description JSON } } "required": [name] }</pre> <p>Para declaraciones de datos:</p> <pre>"type": "array" "maxItems": n "minItems": n "items": { data description JSON }</pre> <p>Donde <i>descripción de datos JSON</i> es la representación del esquema JSON de la descripción de datos PL/I.</p>

Esquema JSON para correlación PL/I:

El programa de utilidad DFHJS2LS soporta correlaciones entre esquemas JSON y estructuras de datos PL/I. Puesto que el compilador de Enterprise PL/I y los compiladores de PL/I más antiguos difieren, se admiten dos opciones de lenguaje: PLI-ENTERPRISE y PLI-OTHER.

Reglas para la correlación de nombres de elemento del esquema para PL/I

Los asistentes de CICS generan nombres exclusivos y válidos para variables PL/I de los nombres de elemento del esquema utilizando las reglas siguientes:

1. Caracteres que no sean A-Z, a-z, 0-9, @, #, _ o \$ se sustituyen por ' X '.

Por ejemplo, monthly-total pasa a ser monthlyXtotal.

Puede utilizar el parámetro **MAPPING-OVERRIDES** para cambiar la forma en la que se manejan otros caracteres. Por ejemplo, si establece el valor **HYPHENS-AS-UNDERSCORES**, ningún guión del esquema JSON se convierte en un guión bajo en lugar de una X. Por ejemplo, monthly-total pasa a ser monthly_total.

2. Los nombres duplicados en el mismo ámbito se convierten en exclusivos añadiendo uno o dos dígitos numéricos a la segunda y siguientes instancias del nombre.

Por ejemplo, tres instancias de year se convierten en year , year1 y year2.

3. Un esquema JSON especifica que una variable tiene una cardinalidad variable si tiene un valor "type" de "array" , y las palabras claves "minItems" y "maxItems" se omiten o tienen diferentes valores. Si el esquema especifica que la variable tiene una cardinalidad variable, los nombres de campo se crean con los sufijos "_cont" y "_num".

Para obtener más información, consulte el apartado "Matrices de variables de elementos en DFHJS2LS" en la página 283.

4. Un esquema JSON especifica que una variable es opcional si no aparece en la matriz de palabra clave "required" que está asociada al tipo "object" del esquema JSON delimitado. Para los campos opcionales, se genera un campo adicional con un sufijo _num añadido al nombre de elemento. En el tiempo de ejecución, es cero para indicar que falta el valor de los datos JSON, y no cero si el valor está presente en los datos JSON.
5. Los nombres de campo están limitados a 31 caracteres. Si un nombre generado, incluidos prefijos y sufijos, supera esta longitud, el nombre del elemento se trunca.

La longitud total del nombre resultante es de 31 caracteres o menos.

Reglas para la correlación de tipos de esquema para PL/I

DFHJS2LS correlaciona valores de tipos de esquema con tipos de datos PL/I según la tabla siguiente. También tenga en cuenta los siguientes puntos:

- Si el parámetro **MAPPING-LEVEL** se establece en 1.2 o superior y el parámetro **CHAR-VARYING** se establece en NULL, los datos de caracteres de longitud variable se correlacionan con series terminadas en nulo y se asigna un carácter adicional al terminador en nulo.
- Si el parámetro **MAPPING-LEVEL** se establece en 1.2 o superior y el parámetro **CHAR-VARYING** no se especifica, de forma predeterminada, los datos de caracteres de longitud variable se correlacionan con un tipo de datos VARYINGZ para Enterprise PL/I y el tipo de datos VARYING para Otros PL/I.

- Los datos binarios de longitud variables se correlacionan con un tipo de datos VARYING si es menor de 32 768 bytes y con un contenedor si es mayor de 32 768 bytes.

Palabra clave de esquema JSON	Descripción de datos PL/I
<p>Todo de:</p> <pre>"type": "null" "type": [] "enum": [] "allOf" "anyOf" "noneOf" "not"</pre>	No soportado
<pre>"\$schema": "http://json-schema.org/draft-04/schema#"</pre>	Esta palabra clave se ignora, pero se supone que es compatible con la especificación de esquema borrador 04 JSON.
<pre>"title": "same text" "description": "more text"</pre>	No se tienen en cuenta estas palabras clave.
<pre>"format": "<predefined values>"</pre>	La palabra clave "format" se utiliza para modificar la estructura generada o el valor de tiempo de ejecución. Consulte la siguiente información para ver el uso soportado de "format".
<pre>"type": "array", "items": {<JSON Sub-schema>}, "additionalItems": false, "maxItems": m , "minItems": n</pre>	<p>La única forma de matriz JSON soportada actualmente es un número repetido de valores del mismo tipo. El <JSON Sub-schema> debe definir un "type" soportado, pero ese "type" no puede ser "array". Se trata de una restricción sobre la estructura de lenguaje generada.</p> <p>Se supone que los "additionalItems" son falsos y no se soporta ningún otro valor.</p> <p>Si "minItems" y "maxItems" están presentes, pero no iguales, la matriz se trata como una cardinalidad fija, de lo contrario, se trata como una cardinalidad variable. Consulte "Matrices de variables de elementos en DFHJS2LS" en la página 283.</p>
<pre>"type": "array", "uniqueItems": true</pre>	"uniqueItems" no se soportan con matrices JSON. El <JSON Sub-schema> debe definir un "type" soportado, pero ese "type" no puede ser "array". Se trata de una restricción sobre la estructura de lenguaje generada.

Palabra clave de esquema JSON	Descripción de datos PL/I
<pre>"type": "object", "additionalProperties": false, "properties": { ["<element name>": {<JSON Sub-schema> } [,]]* } "required": [["<element name>" [,]]*]</pre>	<p>La única forma de objeto JSON soportado actualmente es un conjunto fijo de elementos con nombre.</p> <p>Esto generará una estructura (o subestructura) utilizando los nombres de elemento.</p> <p>Se supone que las "additionalProperties" son false, y no se soporta ningún otro valor.</p> <p>Cualquier elemento del objeto "properties" se considera "optional" si no está en la matriz "required" o si no existe la matriz "required". A un elemento "optional" se le proporciona una ordinalidad variable de cero para X; donde X es 1 o el número máximo de elementos de la matriz, donde el elemento se define como una matriz. Consulte "Matrices de variables de elementos en DFHJS2LS" en la página 283.</p>
<pre>"type": "object", "maxProperties": m, "minProperties": n, "patternProperties": {}, "dependencies":</pre>	<p>No se soporta ninguna de estas palabras claves con objetos JSON.</p>
<pre>"type": "string" "maxLength": m "pattern": "regular expression", "minLength": l</pre>	<p>char[z]</p> <p>donde el valor de z se basa en m , pero depende de los valores del parámetro CHAR-VARYING.</p> <p>m se basa en la palabra clave "maxLength" y se trata como una serie de longitud fija.</p> <p>Las restricciones "pattern" y "minLength" se pasan a través de la estructura de lenguaje sólo como un comentario.</p>
<pre>"type": "string" "maxLength": m</pre>	<p>Cuando CCSID=1200 está en el nivel de correlación 4.0 y superior:</p> <p>WIDECHAR(z)</p> <p>donde el valor de z se basa en m , pero depende de los valores del parámetro CHAR-VARYING.</p> <p>m se basa en la palabra clave "maxLength" y se trata como una serie de longitud fija.</p>

Palabra clave de esquema JSON	Descripción de datos PL/I
<pre>"*name*":{ "type":"string", "format":"date-time" }</pre>	<p>FIXED DECIMAL (15,0)</p> <p>Todo soportado cuando DATETIME=PACKED15</p>
<pre>"*name*":{ "type":"string", "format":"uri" }</pre>	<p>CHAR(<i>m</i>)</p> <p>donde <i>m</i> se basa en la palabra clave "maxLength" y se trata como una serie de longitud fija.</p> <p>Cuando CCSID=1200 está en el nivel de correlación 4.0 y superior:</p> <p>WIDECHAR(<i>m</i>)</p> <p>donde <i>m</i> se basa en la palabra clave "maxLength" y se trata como una serie de longitud fija.</p>
<pre>"*name*":{ "type":"string", "format":"base64Binary" }</pre>	<p>CHAR(<i>m</i>)</p> <p>donde <i>m</i> se basa en la palabra clave "maxLength".</p>
<pre>"*name*":{ "type":"string", "format":"hexBinary" }</pre>	<p>CHAR(<i>m</i>)</p> <p>donde <i>m</i> se basa en la palabra clave "maxLength".</p>
<pre>"*name*":{ "type":"string", "format":"<predefined>" }</pre>	<p>CHAR(<i>m</i>) donde <i>m</i> se basa en la palabra clave "maxLength" y se trata como una serie de longitud fija, y <predefined> es una de: <i>email</i> , <i>hostname</i> , <i>ipv4</i> o <i>ipv6</i>. Se pasa un "pattern" relevante al comentario.</p> <p>Cuando CCSID=1200 está en el nivel de correlación 4.0 y superior:</p> <p>WIDECHAR(<i>m</i>)</p> <p>donde <i>m</i> se basa en la palabra clave "maxLength" y se trata como una serie de longitud fija, y donde <predefined> es uno de: <i>email</i> , <i>hostname</i> , <i>ipv4</i> o <i>ipv6</i> . Se utiliza un "pattern" relevante y se pasa al comentario.</p>

Palabra clave de esquema JSON	Descripción de datos PL/I
"type": "boolean"	<p>Nivel de correlación 1.1 e inferior:</p> <p>Enterprise PL/I SIGNED FIXED BINARY (7)</p> <p>Otros PL/I FIXED BINARY (7)</p> <p>Nivel de correlación 1.2 y superior:</p> <p>Enterprise PL/I BIT(7) BIT(1)</p> <p>Otros PL/I BIT(7) BIT(1)</p> <p>donde se proporciona BIT(7) para la alineación y BIT(1) contiene el valor correlacionado booleano.</p>
"type": "integer", "maxExclusive": true, "minExclusive": true, "multipleOf": n	<p>Las restricciones "maxExclusive" y "minExclusive" se pasan a la estructura de lenguaje sólo como un comentario.</p> <p>"multipleOf" se ignora.</p>
"type": "integer", minimum: -128, maximum: 127	<p>Enterprise PL/I SIGNED FIXED BINARY (7)</p> <p>Otros PL/I FIXED BINARY (7)</p>
"type": "integer", minimum: 0, maximum: 255	<p>Enterprise PL/I UNSIGNED FIXED BINARY (8)</p> <p>Otros PL/I FIXED BINARY (8)</p>
"type": "integer", minimum: -32768, maximum: 32767	<p>Enterprise PL/I SIGNED FIXED BINARY (15)</p> <p>Otros PL/I FIXED BINARY (15)</p>
"type": "integer", minimum: 0, maximum: 65535	<p>Enterprise PL/I UNSIGNED FIXED BINARY (16)</p> <p>Otros PL/I FIXED BINARY (16)</p>
"type": "integer", minimum: -2147483648, maximum: 2147483647	<p>Enterprise PL/I SIGNED FIXED BINARY (31)</p> <p>Otros PL/I FIXED BINARY (31)</p>

Palabra clave de esquema JSON	Descripción de datos PL/I
"type": "integer", minimum: 0, maximum: 4294967295	<p>Nivel de correlación 1.1 e inferior:</p> <p>Enterprise PL/I UNSIGNED FIXED BINARY(32)</p> <p>Nivel de correlación 1.2 y superior:</p> <p>Enterprise PL/I CHAR(<i>y</i>) donde <i>y</i> es una longitud fija menor de 16 MB.</p> <p>Todos los niveles de correlación:</p> <p>Otros PL/I BIT(64)</p>
"type": "integer", minimum: -9223372036854775808, maximum: 9223372036854775807	<p>Nivel de correlación 1.1 e inferior:</p> <p>Enterprise PL/I SIGNED FIXED BINARY(63)</p> <p>Nivel de correlación 1.2 y superior:</p> <p>Enterprise PL/I CHAR(<i>y</i>) donde <i>y</i> es una longitud fija menor de 16 MB.</p> <p>Todos los niveles de correlación:</p> <p>Otros PL/I BIT(64)</p>
"type": "integer", minimum: 0, maximum: 18446744073709551615	<p>Nivel de correlación 1.1 e inferior:</p> <p>Enterprise PL/I UNSIGNED FIXED BINARY(64)</p> <p>Nivel de correlación 1.2 y superior:</p> <p>Enterprise PL/I CHAR(<i>y</i>) donde <i>y</i> es una longitud fija menor de 16 MB.</p> <p>Otros PL/I BIT(64)</p>
"type": "number" "description": "decimal"	FIXED DECIMAL(<i>n</i> , <i>m</i>)

Palabra clave de esquema JSON	Descripción de datos PL/I
<pre>"type": "number" "description": "float"</pre>	<p>Niveles de correlación 1.0 y 1.1:</p> <ul style="list-style-type: none"> CHAR(32) <p>Nivel de correlación 1.2 y superior:</p> <p>Enterprise PL/I DECIMAL FLOAT(6) HEXADEC</p> <p>Otros PL/I DECIMAL FLOAT(6)</p> <p>Nota: La representación de datos de coma flotante hexadecimal (HFP) de IBM no es exactamente lo mismo que la representación IEEE-754-1985 utilizada para JSON. Algunos valores podrían no convertirse de forma exacta de una representación a la otra. Algunos valores sumamente grandes o pequeños podrían no ser válidos para los tipos de datos flotantes. Algunos valores podrían perder precisión cuando se convierten a o desde la representación HFP. Si las conversiones precisas son importantes, considere sustituir el uso de tipos de datos DECIMAL FLOAT por alternativas de precisión fija.</p>
<pre>"type": "number" "description": "double"</pre>	<p>Niveles de correlación 1.0 y 1.1:</p> <ul style="list-style-type: none"> CHAR(32) <p>Nivel de correlación 1.2 y superior:</p> <p>Enterprise PL/I DECIMAL FLOAT(16) HEXADEC</p> <p>Otros PL/I DECIMAL FLOAT(16)</p> <p>Nota: La representación de datos de coma flotante hexadecimal (HFP) de IBM no es exactamente lo mismo que la representación IEEE-754-1985 utilizada para JSON. Algunos valores podrían no convertirse de forma exacta de una representación a la otra. Algunos valores sumamente grandes o pequeños podrían no ser válidos para los tipos de datos flotantes. Algunos valores podrían perder precisión cuando se convierten a o desde la representación HFP. Si las conversiones precisas son importantes, considere sustituir el uso de tipos de datos DECIMAL FLOAT por alternativas de precisión fija.</p>

Nota: CICS no puede transformar valores enteros mayores que el valor máximo para una serie larga firmada ($2^{63} - 1$) a menos que esté encerrada entre comillas.

Nota: Los valores mínimo y máximo especificados en el esquema para tipos numéricos se utilizan únicamente para correlacionar con un tipo de datos PL/I. Los datos no se validan con estos valores durante el tiempo de ejecución.

Matrices de variables de elementos en DFHJS2LS

JSON puede incluir matrices de distintos números de elementos. En general, los esquemas JSON que contienen varios números de elementos no se correlacionan de forma eficiente en una única estructura de datos de lenguaje de alto nivel. CICS utiliza correlaciones basadas en contenedores o correlaciones en línea para gestionar varios elementos en datos JSON.

Una matriz con varios elementos se representa en el esquema JSON utilizando las palabras clave `minItems` y `maxItems` en el esquema con el valor "type" de "array" :

- La palabra clave `minItems` especifica el número de veces mínimo que se puede dar un elemento. Puede tener un valor de 0 o un entero positivo. El valor predeterminado es 0.
- La palabra clave `maxItems` especifica el número de veces máximo que se puede dar un elemento. Puede tener un valor de entero positivo mayor o igual al valor de la palabra clave `minItems`.
- Si falta la palabra clave `maxItems`, significa que la matriz no está limitada.

Se puede indicar un campo opcional por medio de una matriz de variables "maxItems":1 . Por ejemplo, una serie de 8 bytes opcional denominada "component" :

```
"properties":{
  "component": {
    "type":"array",
    "maxItems":1,
    "items": {
      "type": "string",
      "maxLength": 8
    }
  },
  "required": ["component"]
}
```

Se puede obtener el mismo efecto no incluyendo el nombre de campo en el valor de palabra clave "required":

```
"properties":{
  "component": {
    "type": "string",
    "maxLength": 8
  }
}
```

En general, los esquemas JSON que contienen varios números de elementos no se correlacionan de forma eficiente en una única estructura de datos de lenguaje de alto nivel. Para gestionar estos casos, CICS utiliza una serie de estructuras de datos conectadas que se pasan al programa de aplicación en una serie de contenedores. Estas estructuras se utilizan como entrada y salida de la aplicación:

- Cuando CICS transforma datos JSON en datos de aplicación, llena estas estructuras con los datos de aplicación y la aplicación los lee.
- Cuando CICS transforma los datos de aplicación en datos JSON, lee los datos de aplicación en las estructuras que ha estado llenando la aplicación.

Los ejemplos siguientes ilustran el formato de estas estructuras de datos. Estos ejemplos utilizan una matriz de campos simples de 8 bytes. Sin embargo, el modelo soporta matrices de tipos de datos complejos y matrices de tipos de datos que contienen otras matrices.

Ejemplo 1. Número fijo de elementos

Este ejemplo ilustra un elemento que se produce tres veces exactamente:

```
"properties":{
  "component": {
    "type": "array",
    "maxItems": 3,
    "minItems": 3,
    "items": {
      "type": "string",
      "maxLength": 8
    }
  }
},
"required": ["component"]
```

En este ejemplo, el número de veces que se da un elemento se conoce de antemano, por lo tanto, se puede representar como una matriz de longitud fija en una sencilla declaración COBOL o el equivalente en otros lenguajes.

```
05 component PIC X(8) OCCURS 3 TIMES
```

Ejemplo 2. Número de elementos variable en el nivel de correlación 2 e inferior

Este ejemplo ilustra un elemento obligatorio que puede encontrarse de una a cinco veces:

```
"properties":{
  "component": {
    "type": "array",
    "maxItems": 5,
    "minItems": 1,
    "items": {
      "type": "string",
      "maxLength": 8
    }
  }
},
"required": ["component"]
```

La estructura de datos principal contiene una declaración de dos campos. Cuando CICS transforma los datos JSON en datos binarios, el primer campo component-num contiene el número de veces que aparece el elemento en los datos JSON, y el segundo campo, component-cont , contiene el nombre de un contenedor:

```
05 component-num PIC S9(9) COMP-5
05 component-cont PIC X(16)
```

Una segunda estructura de datos contiene la declaración del elemento:

```
01 DFHJS-component
02 component PIC X(8)
```

Debe examinar el valor de component-num , que contendrá un valor en el rango de 1 a 5, para averiguar cuántas veces se da el elemento. El contenido del elemento

está en el contenedor denominado `component-cont`; el contenedor tiene una matriz de elementos, donde cada elemento está correlacionado por la estructura de datos `DFHJS-component`.

Si `minItems="0"` o falta y `maxItems="1"`, el elemento es opcional. Para procesar la estructura de datos en el programa de aplicación, debe examinar el valor de `component-num`:

- Si es cero, el mensaje no tiene elemento de componente y el contenido de `component-cont` no está definido.
- Si es uno, el elemento de componente está en el contenedor denominado `component-cont`.

El contenido del contenedor está correlacionado por la estructura de datos `DFHJS-component`.

Nota: Si los datos JSON constan de un único elemento recurrente, `DFHJS2LS` genera dos estructuras de lenguaje. La estructura de lenguaje principal contiene el número de elementos en la matriz y el nombre del contenedor que tiene la matriz de elementos. La segunda estructura de lenguaje correlaciona una única instancia de elemento recurrente.

Ejemplo 3. Número variable de elementos a nivel de correlación 2.1 y superior

En el nivel de correlación 2.1 y superior, puede utilizar el parámetro **INLINE-MAXOCCURS-LIMIT** en los asistentes de CICS. El parámetro **INLINE-MAXOCCURS-LIMIT** especifica la forma en la que se manejan varios elementos. Las opciones de correlación para un número variado de elementos son correlaciones basadas en contenedores, descritas en “Ejemplo 2. Número de elementos variable en el nivel de correlación 2 e inferior” en la página 285, o correlaciones en línea. El valor de este parámetro puede ser un entero positivo en el rango de 0 a 32767:

- El valor predeterminado de **INLINE-MAXOCCURS-LIMIT** es 1, que garantiza que los elementos opcionales están correlacionados en línea.
- Un valor de 0 para el parámetro **INLINE-MAXOCCURS-LIMIT** impide la correlación en línea.
- Si `maxItems` es menor o igual al valor de **INLINE-MAXOCCURS-LIMIT**, se utiliza la correlación en línea.
- Si `maxItems` es mayor que el valor de **INLINE-MAXOCCURS-LIMIT**, se utiliza la correlación basada en contenedores.

La correlación de un número variado de elementos en línea da como resultado la generación de una matriz, como sucede con el ejemplo de aparición fija anterior, y un contador. El campo `component-num` indica cuántas instancias del elemento están presentes y si están señaladas por una matriz. Para el ejemplos mostrado en “Ejemplo 2. Número de elementos variable en el nivel de correlación 2 e inferior” en la página 285, cuando **INLINE-MAXOCCURS-LIMIT** es menor o igual a 5, la estructura de datos generada es como ésta:

```
05 component-num PIC S9(9) COMP-5 SYNC.  
05 component OCCURS 5 PIC X(8).
```

El primer campo, `component-num`, es idéntico a la salida para el ejemplo de correlación basada en contenedores de la sección anterior. El segundo campo contiene una matriz de longitud 5 que es lo suficientemente larga para contener el número máximo de elementos que puede generar.

La correlación en línea difiere de la correlación basada en contenedores, que almacena el número de apariciones del elemento y el nombre del contenedor donde están colocados los datos, porque almacena todos los datos en el contenedor actual. Almacenar los datos en el contenedor actual normalmente mejorará el rendimiento y hará que la correlación en línea sea preferible.

Ejemplo 4. Matrices de variables anidadas

Los esquemas JSON complejos pueden contener elementos recurrentes variables, que a su vez contienen elementos recurrentes variables. En este caso, la estructura descrita se amplía más allá de los dos niveles descritos en los ejemplos.

Este ejemplo ilustra un elemento opcional llamado "component2" que se anida en un elemento obligatorio llamado "component1", donde el elemento obligatorio se puede dar de una a cinco veces:

```
"properties":{
  "component1": {
    "type": "array",
    "maxItems": 5,
    "minItems": 1,
    "items": {
      "type": "object",
      "properties":{
        "component2":{
          "type": "string",
          "maxLength": 8
        }
      },
      "required": ["component2"]
    }
  },
  "required": ["component1"]
}
```

La estructura de datos de nivel superior es exactamente la misma que en los ejemplos anteriores:

```
05 component1-num PIC S9(9) COMP-5
05 component1-cont PIC X(16)
```

Sin embargo, la segunda estructura de datos contiene estos elementos:

```
01 DFHJS-component1
02 component2-num PIC S9(9) COMP-5
02 component2-cont PIC X(16)
```

Una estructura de tercer nivel contiene estos elementos:

```
01 DFHJS-component2
02 component2 PIC X(8)
```

El número de apariciones del elemento más externo "component1" está en component1-num.

El contenedor nombrado en component1-cont contiene una matriz con varias instancias de la segunda estructura de datos DFHJS-component1.

Cada instancia de component2-cont nombra un contenedor diferente, cada uno de los cuales contiene la estructura de datos correlacionada por la estructura de tercer nivel DFHJS-component2.

Para ilustrar esta estructura, tenga en cuenta el fragmento de datos JSON que coincide con el ejemplo:

```

{"component1":
[
{
"component2": "string1"
},
{
"component2": "string2"
},
]
}

```

"component1" se da tres veces. Las dos primeras contienen una instancia de "component2" ; la tercera instancia no.

En la estructura de datos de nivel superior, component1-num contiene un valor de 3. El contenedor nombrado en component1-cont tiene tres instancias de DFHJS-component1 :

1. En la primera, component2-num tiene un valor de 1, y el contenedor nombrado en component2-cont contiene *string1*.
2. En la segunda, component2-num tiene un valor de 1, y el contenedor nombrado en component2-cont contiene *string2*.
3. En la tercera, component2-num tiene un valor de 0, y el contenido de component2-cont no está definido.

En esta instancia, la estructura de datos completa se representa por medio de cuatro contenedores en total:

- La estructura de datos raíz en el contenedor DFHJS-DATA.
- El contenedor nombrado en component1-cont.
- Los dos contenedores nombrados en las dos primeras instancias de component2-cont.

Estructuras opcionales y la palabra clave required

Las estructuras de datos se definen por el esquema JSON "type" de "object". Los esquemas relacionan nombres de campos con tipos individuales utilizando el objeto proporcionado por la palabra clave "properties". El requisito para que estos campos sean parte de los datos JSON descritos por el esquema JSON se controla por medio de la matriz proporcionada por la palabra clave "required". Esta matriz lista todos los nombres de campo que deben estar presentes en los datos JSON. Por lo tanto, los campos opcionales están representados por su ausencia de la matriz, o cuando la matriz no puede estar vacía, la ausencia de la palabra clave "required". En este caso, todos los campos son opcionales.

Los campos opcionales se tratan como matriz variable de 0 o 1 elementos. Esto añade un campo adicional con el sufijo "-num" añadido al nombre de elemento. Si la longitud total es mayor de 28 caracteres, el nombre de elemento se trunca. En el tiempo de ejecución, no será cero para indicar que el valor estaba presente en los datos JSON y cero si el valor no lo está.

Este ejemplo muestra dos campos, uno necesario llamado "required-structure" y el otro opcional llamado "optional-structure" :

```

{
"type": "object",
"properties": {
"required-structure": {
"type": "string",
"maxLength": 8

```

```

    },
    "optional-structure": {
      "type": "string",
      "maxLength": 8
    }
  },
  "required": [
    "required-structure"
  ]
}

```

La estructura COBOL generada muestra ambos campos, pero la segunda va precedida de "optional-structure-num" que es un recuento de enteros de los elementos, con 0 representando a ninguno y 1 que está presente. El valor se establece para indicar si "optional-structure" contiene datos válidos o no.

```

03 OutputData.
06 required-structure PIC X(8).
06 optional-structure-num PIC S9(9) COMP-5 SYNC.
06 optional-structure PIC X(8).

```

Soporte para UTF-16 en datos de aplicación

Los servicios web de CICS soportan la conversión de datos de aplicación codificados con UTF-16 en XML o JSON y también de XML o JSON en datos de aplicación codificados en UTF-16. Utilice UTF-16 cuando necesite almacenar y procesar datos en varios lenguajes.

Los servicios web JSON y SOAP de CICS soportan la conversión de datos de aplicación codificados con UTF-16 en XML o JSON y también de XML o JSON en datos de aplicación codificados en UTF-16. Unicode es un esquema de codificación de anchura variable que permite a los sistemas manejar datos de forma eficaz.

UTF-16 es una codificación de ancho variable para Unicode, donde cada carácter está representado por 2 o 4 bytes. Los servicios web de CICS soportan CCSID 1200 para datos de aplicación, que es UTF-16 BE (big endian) con IBM Private Use Area. Este comportamiento es coherente con el soporte UTF-16 en todos los lenguajes soportados.

UTF-16 se soporta del nivel de correlación 4.0 en adelante. Puede personalizar cómo se convierten los datos de aplicación utilizando valores de correlación en los asistentes. Para obtener más información sobre niveles de correlación XML, consulte Mapping levels for the CICS assistants. Para obtener más información sobre niveles de correlación JSON, consulte Mapping levels for the CICS JSON assistants.

Nota: UTF-16 requiere más tiempo de proceso y menos eficiencia de almacenamiento que las codificaciones EBCDIC. Además, la mezcla de tipos de codificación provoca un proceso de tiempo de ejecución adicional.

Correlación UTF-16 de esquema XML o JSON a estructuras de lenguaje

El soporte para UTF-16 depende de cómo cree el servicio web. La correlación del esquema XML o JSON a estructuras de lenguaje, también conocida como correlación descendente, tiene las siguientes características. Si UTF-16 está habilitado, todos los campos de texto se correlacionan con campos UTF-16, mientras que los tipos de datos de visualización numérica en COBOL se correlacionan como EBCDIC. Para utilizar UTF-16, establezca el parámetro CCSID de DFHJS2LS, DFHSC2LS o DFHWS2LS en 1200.

Por ejemplo, si el siguiente fragmento de esquema XML estaba presente en WSDL:

```
<xsd:element name="myString" nillable="false">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:maxLength value="20"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

El asistente DFHWS2LS genera el campo siguiente en una estructura de lenguaje COBOL:

```
myString PIC N(
20
) USAGE NATIONAL
```

El parámetro CHAR-MULTIPLIER de los asistentes de servicios web se puede utilizar para especificar la longitud de un campo que generan los asistentes.

CHAR-MULTIPLIER

Cuando utiliza UTF-16, los únicos valores válidos para el parámetro **CHAR-MULTIPLIER** son 2 o 4 , donde 2 es el valor predeterminado.

CHAR-MULTIPLIER = 2 , donde el esquema describe una serie de maxlength x , genera PIC N(x) . El valor **CHAR-MULTIPLIER** = 2 no excluye el uso de pares de sustitución en una serie UTF-16, pero tiene repercusión en el número de caracteres que caben en el campo.

CHAR-MULTIPLIER = 4 genera PIC N($2x$) . Si **CHAR-MULTIPLIER** = 4 , el valor en tiempo de ejecución se llena si la serie incluye caracteres que se pueden expresar en una única unidad de codificación.

Correlación UTF-16 de estructuras de lenguaje a esquema XML o JSON

Correlación de una estructura de lenguaje a un esquema XML o JSON, también conocida como correlación ascendente, se gestiona de forma distinta a la correlación descendente. Si se declara una serie UTF-16 en la estructura de lenguaje, CICS interpreta los datos como datos codificados en UTF-16, de lo contrario, se presupone que los datos están en codificación EBCDIC. El parámetro CCSID para DFHLS2JS, DFHLS2SC o DFHLS2WS indica la codificación de cualquier texto EBCDIC dentro de los datos de aplicación; no debe configurarse para indicar UTF-16.

Los tipos de datos que se interpretan como caracteres UTF-16 son los siguientes: PIC N (n) en COBOL, WIDECHAR(n) en PL/I y char16_t[n] en C y C++.

El parámetro CHAR-USAGE de los asistentes de servicios web se puede utilizar para especificar tipos de datos.

CHAR-USAGE

En COBOL, el tipo de datos nacional, PIC N , se puede utilizar para datos UTF-16 o DBCS. Este valor lo controla la opción del compilador NSYMBOL. Debe establecer el parámetro **CHAR-USAGE** en el asistente en el mismo valor que la opción de compilador NSYMBOL para asegurarse de que los datos se manejan adecuadamente. Se establece normalmente en CHAR-USAGE=NATIONAL cuando utiliza UTF-16.

Si desea mezclar tipos de datos nacionales que contienen datos UTF-16 y DBCS en el mismo libro de copias, puede utilizar los calificadores USAGE NATIONAL o USAGE DISPLAY-1 en campos individuales.

Nota: DFHLS2WS, DFHLS2SC y DFHLS2JS no soportan la cláusula de COBOL GROUP USAGE NATIONAL.

Generación de correlaciones a partir de estructuras de lenguaje

Para crear JavaScript Object Notation (JSON) a partir de datos de aplicación o datos de aplicación a partir de JSON, cree las correlaciones para describir cómo debe transformar CICS los datos y JSON en tiempo de ejecución. Puede comenzar desde cualquier registro de datos de aplicación; por ejemplo, puede comenzar con un COMMAREA, archivo VSAM, cola de almacenamiento temporal o un registro IBM DB2.

Antes de empezar

Antes de crear correlaciones, debe asegurarse de que se cumplen estas condiciones previas:

- Debe tener una estructura de lenguaje que describa el registro de aplicación en un conjunto de datos particionados. La estructura de lenguaje se puede escribir en cualquier lenguaje de alto nivel que soporte el asistente de JSON de CICS: COBOL, PL/I, C y C++.
- Debe configurar el ID de usuario bajo el que se ejecuta DFHLS2JS para utilizar z/OS UNIX.
- El ID de usuario debe tener permiso de lectura para acceder a la estructura de lenguaje y permiso de escritura para poner la salida en los directorios adecuados en z/OS UNIX.
- Debe asignar almacenamiento suficiente para el ID de usuario para que el ID ejecute Java. Puede utilizar cualquier versión soportada de Java.

Acerca de esta tarea

Utilice el asistente de JSON de CICS para crear las correlaciones de datos para el registro de aplicación. El asistente de JSON de CICS crea un paquete CICS y emite mensajes de error sobre elementos no soportados que identifica en la estructura de lenguaje. La información de referencia para el asistente de JSON de CICS lista las restricciones que se aplican a cada lenguaje de alto nivel.

Procedimiento

Ejecute el trabajo por lotes DFHLS2JS. DFHLS2JS tiene parámetros opcionales que selecciona para cumplir los requisitos, como la selección de una página de códigos. Utilice los siguientes parámetros como mínimo:

- Especifique el lenguaje de alto nivel de la estructura de lenguaje en el parámetro **LANG**.
- Especifique el nombre y ubicación de un recurso de paquete en el parámetro **BUNDLE**.
- Especifique el nivel de correlación en el parámetro **MAPPING-LEVEL**. Aunque puede utilizar cualquier nivel de correlación, para obtener las opciones de correlación más avanzadas utilice el nivel de correlación más reciente.

- Especifique la ubicación y la página de códigos de las estructuras de lenguaje que describen el registro de aplicación en los parámetros **PDSMEM** y **PDSACP**.
- Especifique el nombre y ubicación del archivo de esquema JSON **.json** en el parámetro **JSON-SCHEMA**. Si el archivo no existe, DFHLS2JS crea el esquema JSON, pero no la estructura de directorio.
- Especifique el nombre que se utiliza para el recurso de paquete JSONTRANSFRM. Este nombre lo utilizan las aplicaciones para identificar las correlaciones JSON.

El trabajo por lotes crea una estructura de directorio de paquete en z/OS UNIX. El directorio de paquetes tiene un subdirectorio META-INF que contiene el manifiesto de paquete. El trabajo por lotes también crea un esquema JSON y un enlace JSON en el paquete, utilizando los nombres de archivo que ha especificado para los parámetros **JSONTRANSFRM** y **JSON-SCHEMA**.

Instale el recurso BUNDLE que especifica este enlace JSON. El recurso de paquete JSONTRANSFRM crea dinámicamente un recurso JSONTRANSFRM, que define la ubicación del archivo de enlace y esquema JSON. Puede utilizar las vistas operativas Paquete de CICS Explorer y Partes del paquete para comprobar el estado de los recursos BUNDLE instalados.

Resultados

Se genera un paquete de CICS que contiene una transformación JSON.

El ejemplo siguiente muestra DFHLS2JS con el conjunto de parámetros mínimo especificado.

```
//LS2JS JOB 'accounting information',name,MSGCLASS=A
// SET QT=''
//JCLLIB JCLLIB ORDER=FPHLQ.SDFHMOBI
//JAVAPROG EXEC DFHLS2JS,
//INPUT.SYSUT1 DD *
LANG=COBOL
BUNDLE=/u/exampleapp/bundle/test1
LOGFILE=/u/exampleapp/jsbind/example.log
MAPPING-LEVEL=3.0
PDSLIB=//CICSHLQ.SDFHSAMP
PDSMEM=CPYBK2
JSONTRANSFRM=example.jsbind
JSON-SCHEMA=/u/exampleapp/example.json
/*
```

Qué hacer a continuación

Escriba un programa de aplicación para transformar los datos de aplicación en JSON y viceversa. Puede utilizar las mismas correlaciones para ambas transformaciones.

Generación de correlaciones a partir de un esquema JSON

Para crear datos de aplicación desde JavaScript Object Notation (JSON) o JSON desde datos de aplicación, cree las correlaciones para describir cómo debe transformar CICS los datos y JSON en tiempo de ejecución. Puede empezar con un esquema JSON. Una vez que se generan la estructura de lenguaje y la correlación, puede desarrollar la aplicación CICS utilizando la estructura de lenguaje y transformar JSON en datos de aplicación y viceversa.

Antes de empezar

Antes de crear correlaciones, debe asegurarse de que se cumplen estas condiciones previas:

- Debe disponer del esquema JSON que describa un registro de JSON.
- Debe configurar el ID de usuario bajo el que se ejecuta DFHJS2LS para utilizar z/OS UNIX.
- El ID de usuario debe tener permiso de lectura para acceder al esquema JSON y escribir el permiso para colocar la salida en los directorios adecuados en z/OS UNIX.
- Debe asignar almacenamiento suficiente para el ID de usuario para que el ID ejecute Java. Puede utilizar cualquier versión soportada de Java.

Acerca de esta tarea

Utilice el asistente de JSON de CICS para crear las correlaciones de datos para el registro de aplicación. El asistente de JSON de CICS crea un paquete CICS y emite mensajes de error sobre elementos no soportados que identifica en la estructura de lenguaje. La información de referencia para el asistente de JSON de CICS lista las restricciones que se aplican a cada lenguaje de alto nivel. Para obtener más información, consulte el apartado “DFHJS2LS: Conversión de esquema JSON a un lenguaje de alto nivel para interfaz enlazable” en la página 419.

Procedimiento

Ejecute el trabajo por lotes DFHJS2LS. DFHJS2LS tiene parámetros opcionales que selecciona para cumplir los requisitos, como la selección de una página de códigos. Utilice los siguientes parámetros como mínimo:

- Especifique el nombre y ubicación de un recurso de paquete en el parámetro **BUNDLE**.
- Especifique el nombre y ubicación del archivo de esquema JSON en el parámetro **JSON-SCHEMA**.
- Especifique el nivel de correlación en el parámetro **MAPPING-LEVEL**. Aunque puede utilizar cualquier nivel de correlación, para obtener las opciones de correlación más avanzadas utilice el nivel de correlación más reciente.
- Especifique el lenguaje de alto nivel para la estructura de lenguaje generada en el parámetro **LANG**.
- Especifique la ubicación y la página de códigos de las estructuras de lenguaje que describen el registro de aplicación en los parámetros **PDSMEM** y **PDSCP**. DFHJS2LS crea la estructura de lenguaje, pero no la estructura de directorios.
- Especifique el nombre que se utiliza para el recurso de paquete JSONTRANSFRM en CICS. Este nombre lo utilizan las aplicaciones para identificar las correlaciones JSON.

El trabajo por lotes crea una estructura de directorio de paquete en z/OS UNIX. El directorio de paquetes tiene un subdirectorio META-INF que contiene el manifiesto de paquete. El trabajo por lotes también crea un enlace JSON y copia el esquema JSON en el directorio de paquetes, utilizando los nombres de archivo que ha especificado para los parámetros **JSONTRANSFRM** y **JSON-SCHEMA**. El trabajo por lotes también crea la estructura de lenguaje en la ubicación especificada en los parámetros **PDSMEM** y **PDSLIB**.

Instale el recurso BUNDLE que especifica este enlace JSON. El recurso de paquete JSONTRANSFRM crea dinámicamente un recurso JSONTRANSFRM, que define la ubicación del archivo de enlace y esquema JSON. Este recurso de

paquete es visible en CICS cuando ve el contenido del paquete instalado utilizando CICS Explorer. No es un recurso CICS normal y no está visible cuando utiliza CEMT o CPSM WUI.

Resultados

Se genera un paquete de CICS que contiene una transformación JSON. Se genera una estructura de lenguaje.

El ejemplo siguiente muestra DFHJS2LS con el mínimo conjunto de parámetros especificado.

```
//JS2LS JOB 'accounting information',name,MSGCLASS=A
// SET QT=''
//JCLLIB JCLLIB ORDER=FPHLQ.SDFHMOBI
//JAVAPROG EXEC DFHJS2LS,
//INPUT.SYSUT1 DD *
LANG=COBOL
BUNDLE=/u/exampleapp/bundle/test2
LOGFILE=/u/exampleapp/jsbind/example.log
MAPPING-LEVEL=3.0
PDSLIB=/u/exampleapp
PDSMEM=CPYBK2
JSONTRANSFRM=example.jsbind
JSON-SCHEMA=/u/exampleapp/example.json
/*
```

Qué hacer a continuación

Escriba un programa de aplicación para transformar los datos de aplicación en JSON y viceversa. Puede utilizar las mismas correlaciones para ambas transformaciones.

Transformación de datos de aplicación en JSON enlazando con DFHJSON

La interfaz enlazable del transformador JSON, DFHJSON, es un programa proporcionado por que se puede invocar para realizar la transformación entre datos de aplicación y JSON. El programa de aplicación puede transformar datos de aplicación en JSON enlazando con DFHJSON.

Antes de empezar

Debe tener un recurso de paquete JSONTRANSFRM habilitado que defina el enlace JSON y el esquema JSON. Si tiene pensado realizar transformaciones utilizando Java, debe tener un servidor JVM Axis2 ya en ejecución.

Acerca de esta tarea

Cree o actualice un programa de aplicación para enlazar con el programa proporcionado por CICS, DFHJSON, para hacer la transformación

Procedimiento

1. El programa de aplicación debe crear un canal, por ejemplo *MyChannelName* y poner los siguientes contenedores en el canal.
 - DFHJSON-DATA
 - DFHJSON-TRANSFRM
 - DFHJSON-JVMSERV (opcional)

Para obtener más información acerca de estos contenedores, consulte JSON transformer linkable interface containers.

2. Utilice el mandato de API **EXEC CICS LINK PROGRAM** para transformar los datos en JSON:

```
EXEC CICS LINK PROGRAM('DFHJSON') CHANNEL('MyChannelName')
```

3. Consiga el contenedor DFHJSON-ERROR y compruebe si se han producido errores durante la transformación.
4. Consiga el contenedor DFHJSON-JSON y haga uso del JSON en la aplicación.
5. Instale la aplicación.

Resultados

Cuando la aplicación ejecuta el mandato **LINK**, CICS comprueba el recurso de paquete JSONTRANSFRM para encontrar las correlaciones en el enlace JSON y transforma los datos binarios de aplicación en JSON utilizando los contenedores en el canal. El JSON se sitúa en el contenedor DFHJSON-JJSON al devolverse. JSON se ajusta al esquema JSON que se define en el recurso de paquete JSONTRANSFRM.

Qué hacer a continuación

También puede utilizar las mismas correlaciones para transformar JSON en datos de aplicación. Para obtener más información, consulte el apartado “Transformación de JSON en datos de aplicación enlazando con DFHJSON” en la página 485.

Transformación de datos de aplicación en JSON utilizando el mandato de API TRANSFORM DATATOJSON

Puede utilizar el mandato de API **TRANSFORM DATATOJSON** en la aplicación para transformar datos de aplicación en JSON.

Antes de empezar

Debe tener un recurso JSONTRANSFRM habilitado que defina el enlace JSON y el esquema JSON. Si tiene pensado realizar transformaciones utilizando Java, debe tener un servidor JVM Axis2 ya en ejecución.

Acerca de esta tarea

La aplicación debe utilizar una interfaz basada en canal.

Procedimiento

1. Cree un canal y ponga dentro del canal un contenedor de entrada que contenga los datos de aplicación que se va a convertir.

Nota: Este canal también tendrá un contenedor de salida que contiene la salida JSON cuando se completa el mandato **TRANSFORM DATATOJSON**. No cree el contenedor de salida antes de emitir **TRANSFORM DATATOJSON** porque el contenedor se crea y se llena como parte del mandato mismo.

2. Utilice el mandato **TRANSFORM DATATOJSON** para transformar datos en JSON. Por ejemplo:

```
EXEC CICS TRANSFORM DATATOJSON CHANNEL(  
  ChannelName  
) INCONTAINER(  
  InpContainerName  
) OUTCONTAINER(  
  OutContainerName
```

```

OutContainerName
) TRANSFORMER(
BundleName
)

```

Resultados

Cuando la aplicación ejecuta el mandato **TRANSFORM DATATOJSON**, CICS comprueba el recurso de paquete JSONTRANSFRM para encontrar las correlaciones en el enlace JSON y transformar los datos binarios de aplicación en JSON utilizando los contenedores en el canal. A la vuelta, JSON se coloca en el contenedor especificado en la opción **OUTCONTAINER** del mandato **TRANSFORM DATATOJSON**. Si la opción se omite, se utiliza DFHJSON-JSON de forma predeterminada. JSON se ajusta al esquema JSON que se define en el recurso de paquete JSONTRANSFRM.

Transformación de JSON en datos de aplicación enlazando con DFHJSON

La interfaz enlazable del transformador JSON, DFHJSON, es un programa proporcionado por que se puede invocar para realizar la transformación entre datos de aplicación y JSON. El programa de aplicación puede transformar JSON en datos de aplicación enlazando con DFHJSON.

Antes de empezar

Debe tener un recurso JSONTRANSFRM habilitado que defina el enlace JSON y el esquema JSON. Si tiene pensado realizar transformaciones utilizando Java, debe tener un servidor JVM Axis2 ya en ejecución.

Acerca de esta tarea

Cree o actualice un programa de aplicación para enlazar con el programa proporcionado por CICS, DFHJSON, para hacer la transformación

Procedimiento

1. Cree un canal, por ejemplo *MyChannelName* y ponga los siguientes contenedores en el canal.
 - DFHJSON-JSON
 - DFHJSON-TRANSFRM
 - DFHJSON-JVMSERVER

Para obtener más información acerca de estos contenedores, consulte JSON transformer linkable interface containers.

2. Utilice el mandato de API **EXEC CICS LINK PROGRAM** para transformar los datos en JSON:


```
EXEC CICS LINK PROGRAM('DFHJSON') CHANNEL('MyChannelName')
```
3. Instale el programa de aplicación.

Resultados

Cuando la aplicación ejecuta el mandato **LINK PROGRAM**, CICS comprueba el recurso JSONTRANSFRM para encontrar las correlaciones en el enlace de JSON y transforma JSON en datos de aplicación utilizando los contenedores en el canal. Los datos de aplicación se colocan en el contenedor de bits DFHJSON-DATA al devolverse.

Qué hacer a continuación

También puede utilizar las mismas correlaciones para transformar datos de aplicación para JSON. Para obtener más información, consulte el apartado “Transformación de datos de aplicación en JSON enlazando con DFHJSON” en la página 483.

Transformación de JSON en datos de aplicación utilizando el mandato de API TRANSFORM JSONTODATA

Puede utilizar el mandato de API **TRANSFORM JSONTODATA** en la aplicación para transformar JSON en datos de aplicación.

Antes de empezar

Debe tener un recurso JSONTRANSFRM habilitado que defina el enlace JSON y el esquema JSON. Si tiene pensado realizar transformaciones utilizando Java, debe tener un servidor JVM Axis2 ya en ejecución.

Acerca de esta tarea

La aplicación debe utilizar una interfaz basada en canal.

Procedimiento

1. Cree un canal y ponga dentro del canal un contenedor de entrada que contiene el JSON que se va a convertir.

Nota: Este canal también tendrá un contenedor de salida que contiene los datos convertidos cuando se completa el mandato **TRANSFORM JSONTODATA**. No cree el contenedor de salida antes de emitir **TRANSFORM JSONTODATA** porque el contenedor se crea y se llena como parte del mandato mismo.

2. Utilice el mandato **TRANSFORM JSONTODATA** para transformar JSON en datos de aplicación. Por ejemplo:

```
EXEC CICS TRANSFORM JSONTODATA CHANNEL(  
  ChannelName  
) INCONTAINER(  
  InpContainerName  
) OUTCONTAINER(  
  OutContainerName  
) TRANSFORMER(  
  BundleName  
)
```

Resultados

Cuando la aplicación ejecuta el mandato **TRANSFORM JSONTODATA**, CICS comprueba el recurso de paquete JSONTRANSFRM para encontrar las correlaciones en el enlace JSON y transformar el JSON en datos binarios de aplicación utilizando los contenedores en el canal. A la vuelta, los datos convertidos se colocan en el contenedor especificado en la opción **OUTCONTAINER** del mandato **TRANSFORM JSONTODATA**. Si la opción se omite, se utiliza DFHJSON-DATA de forma predeterminada.

Creación de una aplicación de cliente de servicio web JSON

Puede escribir un programa de aplicación para invocar un servicio web RESTful utilizando la interfaz enlazable para transformar JSON y, a continuación, utilizar los mandatos WEB para enviarlo al proveedor de servicios remoto.

Antes de empezar

Debe familiarizarse con el uso de la interfaz enlazable para transformar JSON, como se describe en Transformación de datos de aplicación y JSON utilizando la interfaz enlazable, y con las API de EXEC CICS WEB, como se describe en .

Acerca de esta tarea

Como parte de una aplicación CICS, es posible que desee invocar un servicio web RESTful alojado en otro sistema. Para ello, primero debe describir los datos que se van a intercambiar con el servicio remoto. Después, puede escribir un programa de aplicación que utilice la API de EXEC CICS WEB para comunicarse con el servicio remoto con el protocolo HTTP para enviar datos de solicitud al servicio y recibir datos de respuesta. Puede utilizar la interfaz enlazable para transformar los datos de aplicación en JSON para utilizarlos como parte de la solicitud, y transformar la respuesta JSON en datos de aplicación. Puede que algunos servicios no soporten una carga útil para la solicitud y la respuesta.

Procedimiento

1. Defina la interfaz en el servicio remoto.
 - a. Si existe el servicio remoto, compruebe si está disponible el esquema JSON que describe las cargas útiles de solicitud y respuesta. Si no, debe crear uno. A continuación, utilice el asistente de JSON para generar una correlación a una estructura de lenguaje. Para obtener más información, consulte el apartado “Generación de correlaciones a partir de un esquema JSON” en la página 481.
 - b. Si el servicio remoto no existe todavía y desea basar su interfaz en la estructura de datos de aplicación, utilice el asistente de JSON para generar un esquema JSON. A continuación, pase el esquema JSON al desarrollador de aplicación del servicio remoto. Para obtener más información, consulte el apartado “Generación de correlaciones a partir de estructuras de lenguaje” en la página 480.
2. Defina un recurso BUNDLE para un paquete generado por el asistente de JSON e instale el paquete en CICS.
3. Defina un recurso URIMAP para el punto final de servicio remoto e instálelo. Para obtener más información, consulte el apartado URIMAP resources.
4. Cree o actualice un programa de aplicación para invocar el servicio remoto de la siguiente manera:
 - a. Si el servicio remoto requiere una carga útil JSON para la solicitud (por ejemplo, cuando el método HTTP es POST o PUT), utilice la interfaz enlazable para transformar los datos de aplicación en JSON. Para obtener más información, consulte el apartado “Transformación de datos de aplicación en JSON enlazando con DFHJSON” en la página 483.
 - b. Abra una conexión con el servidor donde está alojado el servicio remoto, utilizando el mandato **EXEC CICS WEB OPEN**. Para obtener más información, consulte el apartado WEB OPEN.
 - c. Dependiendo de los requisitos del servicio, es posible que desee codificar el mandato **EXEC CICS WEB WRITE HTTPHEADER** para especificar la cabecera

Contenido-Tipo application/JSON para indicar que se está proporcionando JSON. Para obtener más información, consulte el apartado WEB WRITE HTTPHEADER.

- d. Codifique un mandato **EXEC CICS WEB CONVERSE** para enviar la solicitud al servicio remoto y recibir la respuesta. Especifique la serie de consulta o el cuerpo de solicitud (desde el contenedor DFHJSON-JSON), si es necesario. Si espera una respuesta del servicio, especifique el contenedor DFHJSON-JSON para recibir el JSON de respuesta. Para obtener más información, consulte el apartado WEB CONVERSE.
- e. Si no tiene previsto hacer más solicitudes, codifique un mandato **EXEC CICS WEB CLOSE** para cerrar la conexión. Para obtener más información, consulte el apartado WEB CLOSE.
- f. Compruebe el código de respuesta HTTP que devuelve el mandato **EXEC CICS WEB CONVERSE** y realice la acción adecuada si se produce un error. Por ejemplo, intente la solicitud de nuevo o devuelva un error al usuario.
- g. Si se espera un cuerpo de respuesta del servicio remoto, utilice la interfaz enlazable para transformar el JSON en datos de aplicación. Para obtener más información, consulte el apartado “Transformación de datos de aplicación en JSON enlazando con DFHJSON” en la página 483.

Resultados

Ha creado una aplicación que puede invocar un servicio web RESTful con una carga útil JSON.

Creación de un servicio web JSON

Puede exponer las aplicaciones CICS existentes como servicios web SOAP y crear nuevas aplicaciones CICS para que actúen como solicitantes o proveedores de servicios web.

Antes de empezar

Antes de empezar a crear un servicio web SOAP, realice estas tareas:

1. Configure el sistema CICS para dar soporte a servicios web; consulte “Configuración del sistema CICS para servicios web” en la página 41.
2. Cree la infraestructura necesaria para soportar el despliegue de servicios web; consulte “Creación de la infraestructura de servicios web” en la página 54.
3. Decida si desea utilizar el asistente de servicios web; consulte “Planificación para utilizar servicios web SOAP” en la página 27.

Acerca de esta tarea

El asistente de servicios web de CICS es un programa de utilidad proporcionado que le ayuda a crear los artefactos necesarios para una nueva aplicación de solicitante de servicio o proveedor de servicios web SOAP, o para habilitar una aplicación existente como proveedor de servicios web.

El asistente de servicios web de CICS puede crear un documento WSDL desde una estructura de lenguaje sencilla o una estructura de lenguaje desde el documento WSDL existente; y soporta COBOL, C/C++ y PL/I. También genera información utilizada para habilitar la conversión en tiempo de ejecución automática de los mensajes SOAP para contenedores y COMMAREAs, y viceversa. El soporte de servicios web de CICS utiliza esta información durante el proceso de interconexión.

Cree su servicio web, como se describe en el siguiente procedimiento y valide que funciona correctamente:

Procedimiento

1. Cree un servicio web SOAP de una de las cuatro formas siguientes:
 - Utilice el asistente de servicios web para crear la descripción de servicio web o las estructuras de lenguaje y despléguelas en CICS. Utilice el mandato **PIPELINE SCAN** para crear automáticamente los recursos CICS necesarios.
 - Utilice IBM Developer for z Systems o la API de Java para crear la descripción de servicio web o las estructuras de lenguaje y despléguelas en CICS. Utilice el mandato **PIPELINE SCAN** para crear automáticamente los recursos CICS necesarios.
 - Cree o cambie un programa de aplicación para manejar el XML en los mensajes de entrada y salida, incluida la conversión de datos, y llene los contenedores correctos en la interconexión. Debe crear los recursos de CICS necesarios manualmente.
 - Despliegue una aplicación Axis2 como servicio web.
2. Inicie el servicio web para probar que funciona como esperaba. Si está utilizando el asistente de servicios web para desplegar el servicio web, puede utilizar el mandato **SET WEBSERVICE** para activar la validación. La validación comprueba que los datos se convierten correctamente.

Qué hacer a continuación

Estos pasos se explican de forma más detallada en los temas siguientes.

El asistente de servicios web de CICS

El asistente de servicios web de CICS es un conjunto de programas de utilidad por lotes que puede ayudarle a transformar aplicaciones CICS existente en servicios web y a habilitar aplicaciones CICS para utilizar servicios web proporcionados por proveedores externos. El asistente soporta un despliegue rápido de aplicaciones CICS para utilizarlas en proveedores de servicios y solicitantes de servicio, con un mínimo esfuerzo de programación.

Cuando utiliza el asistente de servicios web para CICS, no tiene que escribir su propio código para analizar mensajes de entrada y para construir mensajes de salida; CICS correlaciona datos entre el cuerpo de un mensaje SOAP y la estructura de datos del programa de aplicación.

El asistente puede crear un documento WSDL desde una estructura de lenguaje sencilla o una estructura de lenguaje desde el documento WSDL existente, y soporta COBOL, C/C++ y PL/I. También genera información utilizada para habilitar la conversión en tiempo de ejecución automática de los mensajes SOAP para contenedores y COMMAREAs, y viceversa.

El asistente de servicios web de CICS consta de dos programas de utilidad:

DFHLS2WS

Genera un archivo de enlace de servicio web desde una estructura de lenguaje. Este programa de utilidad también genera una descripción de servicio web.

DFHWS2LS

Genera un archivo de enlace de servicio web desde una descripción de

servicio web. Este programa de utilidad también genera una estructura de lenguaje que puede utilizar en los programas de aplicación.

Los procedimientos JCL para ejecutar ambos programas están en la biblioteca *hlq.XDFHINST*.

Para obtener más información sobre los programas de utilidad del asistente de servicios web y las correlaciones de datos, consulte los temas siguientes.

DFHLS2WS: Conversión de lenguaje de alto nivel a WSDL

El procedimiento DFHLS2WS genera una descripción de servicio web y un archivo de enlace de servicio web desde una estructura de datos de lenguaje de alto nivel. Puede utilizar DFHLS2WS cuando expone un programa de aplicación CICS como proveedor de servicios.

Las sentencias de control de trabajos para DFHLS2WS, sus parámetros simbólicos, sus parámetros de entrada y sus descripciones y un trabajo de ejemplo le ayudan a utilizar este procedimiento.

Sentencias de control de trabajos para DFHLS2WS

JOB Inicia el trabajo.

EXEC Especifica el nombre de procedimiento (DFHLS2WS).

INPUT.SYSUT1 DD

Especifica la entrada. Los parámetros de entrada se especifican normalmente en el flujo de entrada. Sin embargo, se pueden definir en un conjunto de datos en miembro de un conjunto de datos particionados.

Parámetros simbólicos

Los siguientes parámetros simbólicos se definen en DFHLS2WS:

JAVADIR = *vía de acceso*

Especifica el nombre del directorio Java que utiliza DFHLS2WS. El valor de este parámetro se añade a /usr/lpp/ para crear un nombre de vía de acceso completo de /usr/lpp/ *vía de acceso*.

Generalmente, no es necesario especificar este parámetro. El valor predeterminado es el valor proporcionado para el trabajo de instalación (DFHISTAR) de CICS en el parámetro **JAVADIR**.

PATHPREF = *prefijo*

Especifica un prefijo opcional que amplía la vía de acceso del directorio z/OS UNIX utilizado en otros parámetros. El valor predeterminado es la serie vacía.

Generalmente, no es necesario especificar este parámetro. El valor predeterminado es el valor proporcionado para el trabajo de instalación (DFHISTAR) de CICS en el parámetro **PATHPREF**.

SERVICE = *valor*

Utilice este parámetro sólo cuando el soporte de IBM le indique que lo haga.

TMPDIR = *tmpdir*

Especifica la ubicación de un directorio en z/OS UNIX que DFHLS2WS utiliza como estación de trabajo temporal. El ID de usuario bajo el que se ejecuta el trabajo debe tener permiso de lectura y escritura para este directorio.

El valor predeterminado es /tmp.

TMPFILE = *tmpprefix*

Especifica un prefijo que DFHLS2WS utiliza para construir los nombres de los archivos de espacio de trabajo temporal.

El valor predeterminado es LS2WS.

USSDIR = *vía de acceso*

Especifica el nombre del directorio CICS TS en el sistema de archivos de servicios del sistema UNIX. El valor de este parámetro se añade a */usr/lpp/cicsts/* para crear un nombre de vía de acceso completa de */usr/lpp/cicsts/ path* .

Generalmente, no es necesario especificar este parámetro. El valor predeterminado es el valor proporcionado para el trabajo de instalación (DFHISTAR) de CICS en el parámetro **USSDIR**.

Espacio de trabajo temporal

DFHLS2WS crea estos tres archivos temporales en tiempo de ejecución:

```
tmpdir / tmpprefix .in  
tmpdir / tmpprefix .out  
tmpdir / tmpprefix .err
```

donde:

tmpdir es el valor que se especifica en el parámetro **TMPDIR**.

tmpprefix es el valor que se especifica en el parámetro **TMPFILE**.

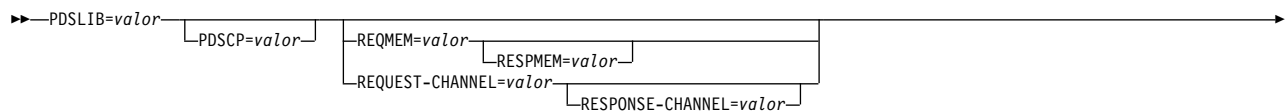
Los nombres predeterminados para los archivos, cuando no se especifica **TMPDIR** y **TMPFILE** son los siguientes:

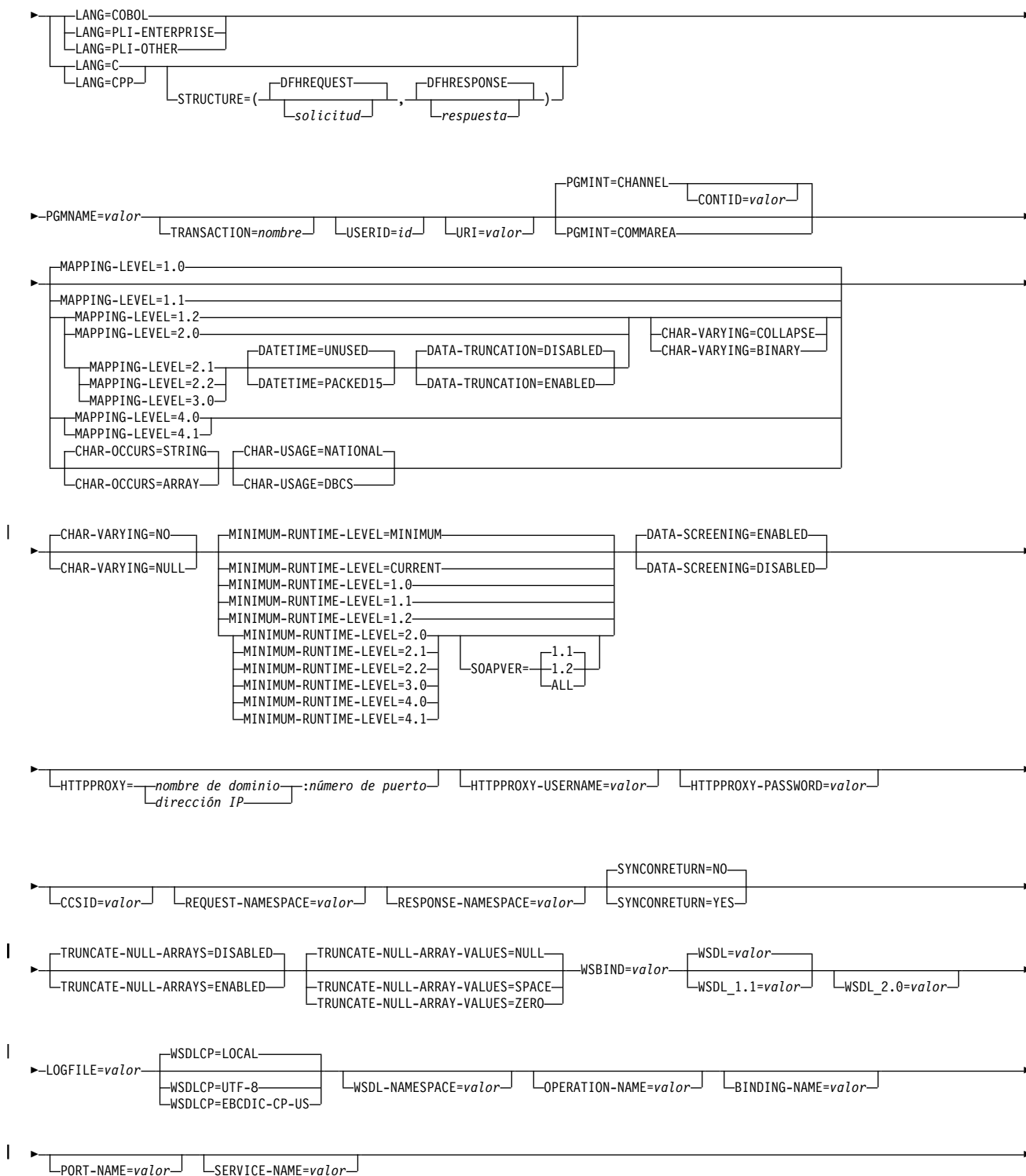
```
/tmp/LS2WS.in  
/tmp/LS2WS.out  
/tmp/LS2WS.err
```

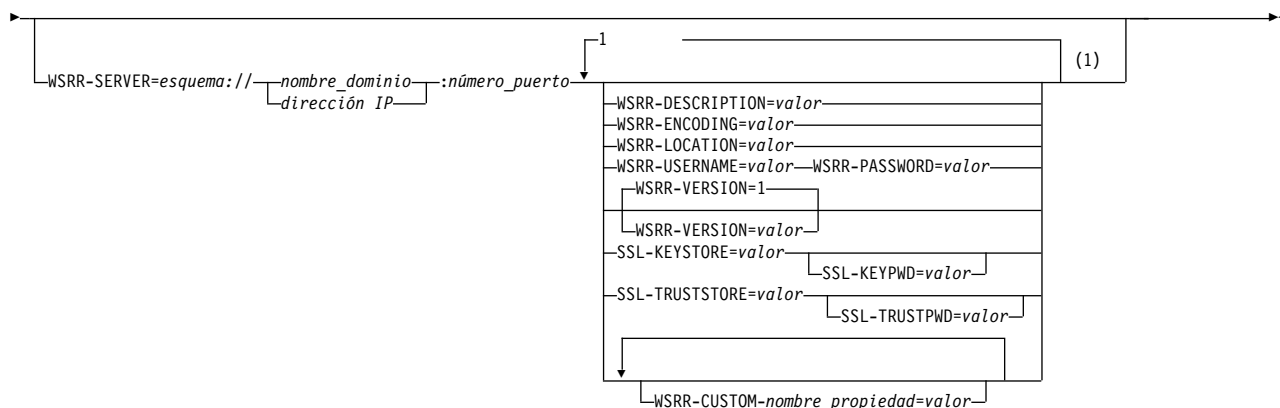
Importante: DFHLS2WS no bloquea el acceso a los archivos z/OS UNIX o los miembros del conjunto de datos. Por lo tanto, si dos o más instancias de DFHLS2WS se ejecutan de forma simultánea y utilizan los mismos archivos de espacio de trabajo temporal, nada impide a un trabajo sobrescribir los archivos de espacio de trabajo mientras otro trabajo los está utilizando, llevando a errores imprevisibles.

Por lo tanto, se aconseja crear un convenio de denominación y procedimientos operativos que eviten esta situación. Por ejemplo, puede utilizar el parámetro simbólico de sistema **SYSUID** para generar nombres de archivo de espacios de trabajo que son exclusivos para un usuario individual. Estos archivos temporales se suprimen antes de la finalización del trabajo.

Parámetros de entrada de DFHLS2WS







Notas:

- 1 Cada uno de los parámetros WSRR que se pueden especificar cuando se establece el parámetro **WSRR-SERVER** sólo se pueden especificar una vez. La excepción a esta regla es el parámetro **WSRR-CUSTOM**, que puede especificar un máximo de 255 veces.

Utilización de parámetros

- Puede especificar los parámetros de entrada en cualquier orden.
- Cada parámetro se debe iniciar en una nueva línea.
- Un parámetro, y su carácter de continuación, si utiliza uno, no deben sobrepasar la columna 72; las columnas 73 a 80 deben contener espacios en blanco.
- Si un parámetro es demasiado largo para ajustarse en una única línea, utilice el carácter de asterisco (*) al final de la línea para indicar que el parámetro continúa en la línea siguiente. Todo, incluidos los espacios antes del asterisco, se considera parte del parámetro. Por ejemplo:

```
WSBIND=wsbinddir*
      /app1
```

es equivalente a

```
WSBIND=wsbinddir/app1
```

- Un carácter # en la primera posición de carácter de la línea es un carácter de comentario. La línea se ignora.
- Una coma en la última posición de carácter de la línea es un separador de línea opcional y se ignora.

Descripciones de parámetros

BINDING-NAME = *valor*

Especifica el nombre de enlace que se utiliza en el documento WSDL generado. Si no se proporciona valor, se genera un nombre de enlace predeterminado utilizando el valor del parámetro **PGMNAME** seguido de "HTTPSoapBinding". Si SOAPVER se establece en ALL, se añade un sufijo "12" al nombre del enlace SOAP 1.2.

CCSID = *valor*

Especifica el CCSID que se utiliza en tiempo de ejecución para codificar datos de caracteres en la estructura de datos de la aplicación. El valor de este parámetro sustituye el valor del parámetro de inicialización del sistema **LOCALCCSID**. El *valor* debe ser un EBCDIC CCSID soportado por los servicios de conversión Java y z/OS (consulte z/OS Unicode Services User's Guide and

Reference). Si no especifica este parámetro, la estructura de datos de aplicación se codifica utilizando el CCSID especificado en el parámetro de inicialización del sistema.

Puede utilizar este parámetro con cualquier nivel de correlación.

CHAR-VARYING = { **NO** | **NULL** | **COLLAPSE** | **BINARY** }

Especifica cómo los campos de carácter de la estructura de lenguaje se correlacionan cuando el nivel de correlación es 1.2 o superior. Un campo de carácter en COBOL es una cláusula Picture de tipo X, por ejemplo PIC(X) 10 ; un campo de carácter en C/C++ es una matriz de caracteres. Puede seleccionar estas opciones:

NO Los campos de carácter se correlacionan con un <xsd:string> y se procesan como campos de longitud fija. La longitud máxima de los datos es igual a la longitud del campo. NO es el valor predeterminado para el parámetro **CHAR-VARYING** para COBOL y PL/I a niveles de correlación 2.0 y anteriores.

Este valor no se aplica a estructuras de lenguaje Enterprise y Otros PL/I.

NULL Los campos de carácter se correlacionan con un <xsd:string> y se procesan como series terminadas en nulo. CICSañade un carácter nulo de terminación cuando se está transformando desde un mensaje SOAP. La longitud máxima de la serie de caracteres se calcula como un carácter menos que la longitud indicada en la estructura de lenguaje. NULL es el valor predeterminado para el parámetro **CHAR-VARYING** para C/C++.

Este valor no se aplica a estructuras de lenguaje Enterprise y Otros PL/I.

COLLAPSE

Los campos de carácter se correlacionan con una <xsd:string> . Los espacios en blanco finales del campo no se incluyen en el mensaje SOAP. El mensaje SOAP de entrada se analiza para eliminar todos los espacios en blanco iniciales, finales e incluidos. COLLAPSE es el valor predeterminado para el parámetro **CHAR-VARYING** para COBOL y PL/I a nivel de correlación 2.1 y en adelante.

Para obtener más información sobre los valores de longitud variable y espacio en blanco, consulte Soporte para valores de longitud variable y espacio en blanco.

BINARY

Los campos de carácter se correlacionan con un <xsd:base64binary> y se procesan como campos de longitud fija. El valor BINARY en el parámetro **CHAR-VARYING** está disponible sólo en niveles de correlación 2.1 y en adelante.

CHAR-OCCURS = { **STRING** | **ARRAY** }

Especifica cómo las matrices de caracteres en la estructura de lenguaje se correlacionan cuando el nivel de correlación es 4.0 o superior. Por ejemplo, PIC X OCCURS 20 . Este parámetro sólo lo utiliza el lenguaje COBOL.

ARRAY

Las matrices de caracteres se correlacionan con una matriz de XML. Esto significa que cada carácter se correlaciona como un elemento XML individual. Este también es el comportamiento en el nivel de correlación 3.0 y anterior.

STRING

Las matrices de caracteres se correlacionan con una serie XML. Esto significa que toda la matriz COBOL se correlaciona como un único elemento XML.

CHAR-USAGE = { NATIONAL | DBCS }

En COBOL, el tipo de datos nacional, PIC N , se puede utilizar para datos UTF-16 o DBCS. Este valor lo controla la opción del compilador NSYMBOL. Debe establecer el parámetro **CHAR-USAGE** en el asistente en el mismo valor que la opción de compilador NSYMBOL para asegurarse de que los datos se manejan adecuadamente. Se establece normalmente en CHAR-USAGE=NATIONAL cuando utiliza UTF-16.

DBCS Los datos de los campos PIC (*n*) se tratan como datos cifrados en DBCS.

NATIONAL

Los datos de los campos PIC (*n*) se tratan como datos cifrados en UTF-16.

CONTID = *valor*

En un proveedor de servicios, especifique el nombre del contenedor que tiene la estructura de datos de nivel superior utilizada para representar un mensaje SOAP.

La longitud del contenedor que CICS pasa al programa de aplicación de destino es mayor que las longitudes del contenedor de solicitud y el contenedor de respuesta.

DATA-SCREENING = { ENABLED | DISABLED }

Especifica si los datos proporcionados por la aplicación se analizan para detectar errores.

ENABLED

Los datos de tiempo de ejecución proporcionados por la aplicación que son incoherentes con la estructura de lenguaje, se tratan como un error y se emite un mensaje DFHPI1010. Se devuelve una respuesta de error a la aplicación.

DISABLED

Los valores de datos de los datos de tiempo de ejecución proporcionados por la aplicación que son incoherentes con la estructura de lenguaje se sustituyen por valores predeterminados. Por ejemplo, un cero sustituye un valor erróneo en un campo numérico. No se emite mensaje DFHPI1010 y se devuelve una respuesta normal a la aplicación. Esta característica se puede utilizar para evitar las respuestas de error INVALID_PACKED_DEC y INVALID_ZONED_DEC generadas desde los campos de salida no inicializados.

DATA-TRUNCATION = { DISABLED | ENABLED }

Especifica si los datos de longitud variable se soportan en una estructura de campo de longitud fija:

DISABLED

Si los datos son menores que la longitud fija que está esperando CICS, CICS rechaza los datos truncados y emite un mensaje de error.

ENABLED

Si los datos son menores que la longitud fija que está esperando CICS, CICS tolera los datos truncados y procesa los datos que faltan como valores nulos

DATETIME = { **UNUSED** | **PACKED15** }

Especifica si los campos ABSTIME potenciales en la estructura de lenguaje de alto nivel se correlacionan como indicaciones de fecha y hora:

PACKED15

Los campos decimales empaquetados de longitud 15 (8 bytes) se tratan como campos CICS ABSTIME y se correlacionan como indicaciones de fecha y hora.

UNUSED

Los campos decimales empaquetados de longitud 15 (8 bytes) no se tratan como indicaciones de fecha y hora.

Este parámetro se puede definir en un nivel de correlación de 3.0.

HTTPPROXY = { *nombre de dominio : número de puerto* | *dirección IP : número de puerto* }

Si el WSDL contiene referencias a otros archivos WSDL ubicados en Internet, y el sistema en el que está ejecutando DFHLS2WS utiliza un servidor proxy para acceder a Internet, especifique el nombre de dominio o dirección IP y el número de puerto del servidor proxy. Por ejemplo:

HTTPPROXY=proxy.example.com:8080

En otros casos, este parámetro no es necesario.

HTTPPROXY-PASSWORD = *valor*

Especifica la contraseña del proxy HTTP que se debe utilizar con **HTTPPROXY-USERNAME** si el sistema en el que está ejecutando DFHLS2WS utiliza un servidor proxy HTTP para acceder a Internet, y el servidor proxy HTTP utiliza la autenticación básica. Puede utilizar este parámetro sólo cuando especifica **HTTPPROXY**.

HTTPPROXY-USERNAME = *valor*

Especifica la el nombre de usuario del proxy HTTP que se debe utilizar con **HTTPPROXY-PASSWORD** si el sistema en el que está ejecutando DFHLS2WS utiliza un servidor proxy HTTP para acceder a Internet, y el servidor proxy HTTP utiliza la autenticación básica. Puede utilizar este parámetro sólo cuando especifica **HTTPPROXY**.

LANG = **COBOL**

Especifica que el lenguaje de programación de la estructura de lenguaje de alto nivel es COBOL.

LANG = **PLI-ENTERPRISE**

Especifica que el lenguaje de programación de la estructura de lenguaje de alto nivel es Enterprise PL/I.

LANG = **PLI-OTHER**

Especifica que el lenguaje de programación de la estructura de lenguaje de alto nivel es un nivel de PL/I distinto a Enterprise PL/I.

LANG = **C**

Especifica que el lenguaje de programación de la estructura de lenguaje de alto nivel es C.

LANG = CPP

Especifica que el lenguaje de programación de la estructura de lenguaje de alto nivel es C++.

LOGFILE = valor

El nombre completo del archivo de z/OS UNIX en el que DFHLS2WS escribe la información de rastreo y registro de actividad. DFHLS2WS crea el archivo pero no la estructura de directorio, si no existe aún.

Normalmente, no utiliza este archivo, pero puede solicitarlo la organización de servicio de IBM si encuentra problemas con DFHLS2WS.

MAPPING-LEVEL = { 1.0 | 1.1 | 1.2 | 2.0 | 2.1 | 2.2 | 3.0 | 4.0 | 4.1 }

Especifica el nivel de correlación que utiliza DFHLS2WS cuando genera el archivo de enlace de servicio web y la descripción de servicio web. Puede seleccionar estas opciones:

- 1.0** Este nivel de correlación es el predeterminado. Indica que el archivo de enlace de servicio web se genera utilizando los niveles de correlación CICS TS 3.1.
- 1.1** Utilice esta correlación para volver a generar un archivo de enlace en este nivel específico.
- 1.2** En este nivel de correlación, puede utilizar el parámetro **CHAR-VARYING** para controlar cómo se procesan las matrices de caracteres en tiempo de ejecución. Las matrices VARYING y VARYINGZ también se soportan en PL/I.
- 2.0** Utilice este nivel de correlación en una región de CICS TS 3.2 o posterior para aprovecharse de las mejoras en las correlaciones entre la estructura de lenguaje y el archivo de enlace de servicios web.
- 2.1** Utilice este nivel de correlación con una región de CICS TS 3.2 que tiene aplicado APAR PK59794 o con cualquier región posterior a CICS TS 3.2. En este nivel, puede beneficiarse de los nuevos valores para el parámetro **CHAR-VARYING**, **COLLAPSE** y **BINARY**. Los campos FILLER en COBOL y los campos * en PL/I se ignoran sistemáticamente en este nivel de correlación, los campos no aparecen en el documento WSDL generado, y queda un hueco adecuado en las estructuras de datos en el tiempo de ejecución.
- 2.2** Utilice este nivel de correlación con una región de CICS TS 3.2 que tiene aplicado APAR PK69738 o con cualquier región posterior a CICS TS 3.2 para beneficiarse de las mejoras de correlación cuando se utiliza DFHWS2LS.
- 3.0** Utilice este nivel de correlación con una región de CICS TS 4.1. En este nivel de correlación, puede crear un servicio web desde una aplicación que utilice contenedores en su interfaz estableciendo los parámetros **REQUEST-CHANNEL** y **RESPONSE-CHANNEL**. También puede correlacionar los campos dateTime con indicaciones de fecha y hora XML estableciendo el parámetro **DATETIME**.
- 4.0** Utilice este nivel de correlación con una región de CICS TS 5.2 o región posterior. En este nivel de correlación, puede utilizar campos de COBOL OCCURS DEPENDING ON y el parámetro **CHAR-OCCURS**.
- 4.1** Utilice este nivel de correlación para el soporte de matriz truncable, con una región de CICS TS V5.2 que tiene aplicado APAR PI67641, o una región de CICS TS V5.3 o posterior.

Para obtener más información sobre los niveles de correlación, consulte Niveles de correlación para los asistentes de CICS

MINIMUM-RUNTIME-LEVEL = { MINIMUM | 1.0 | 1.1 | 1.2 | 2.0 | 2.1 | 2.2 | 3.0 | 4.0 | 4.1 | CURRENT }

Especifica el entorno de ejecución de CICS mínimo en el que se puede desplegar el archivo de enlace de servicio web. Si selecciona un nivel que no coincide con los demás parámetros que ha especificado, recibe un mensaje de error. Puede seleccionar estas opciones:

MINIMUM

El nivel de tiempo de ejecución más bajo posible de CICS se asigna automáticamente a los parámetros determinados que seleccione.

- 1.0** El archivo de enlace de servicios web generado se despliega correctamente en una región de CICS TS 3.1 que no tiene aplicado APARs PK15904 y PK23547. Algunos parámetros no están disponibles en este nivel de tiempo de ejecución.
- 1.1** El archivo de enlace de servicios web generado se despliega correctamente en una región de CICS TS 3.1 que tiene aplicado al menos APAR PK15904. Puede utilizar un nivel de correlación de 1.1 o anterior para el parámetro **MAPPING-LEVEL**. Algunos parámetros no están disponibles en este nivel de tiempo de ejecución.
- 1.2** El archivo de enlace de servicios web generado se despliega correctamente en una región de CICS TS 3.1 que tiene aplicado APAR PK15904 y PK23547. Puede utilizar un nivel de correlación de 1.2 o anterior para el parámetro **MAPPING-LEVEL**. Algunos parámetros no están disponibles en este nivel de tiempo de ejecución.
- 2.0** El archivo de enlace de servicios web generado se despliega correctamente en una región de CICS TS 3.2 o posterior. Puede utilizar un nivel de correlación de 2.0 o anterior para el parámetro **MAPPING-LEVEL**. Algunos parámetros no están disponibles en este nivel de tiempo de ejecución.
- 2.1** El archivo de enlace de servicios web generado se despliega correctamente en una región de CICS TS 3.2 que tiene aplicado APAR PK59794, o en cualquier región posterior a CICS TS 3.2. Puede utilizar un nivel de correlación de 2.1 o anterior para el parámetro **MAPPING-LEVEL**. Algunos parámetros no están disponibles en este nivel de tiempo de ejecución.
- 2.2** El archivo de enlace de servicios web generado se despliega correctamente en una región de CICS TS 3.2 que tiene aplicado APAR PK69738, o en cualquier región posterior a CICS TS 3.2. Con este nivel de ejecución, puede utilizar un nivel de correlación de 2.2 o anterior para el parámetro **MAPPING-LEVEL**. Algunos parámetros no están disponibles en este nivel de tiempo de ejecución.
- 3.0** El archivo de enlace de servicios web generado se despliega correctamente en una región de CICS TS 4.1 o posterior. Con este nivel de ejecución, puede utilizar un nivel de correlación de 3.0 o anterior para el parámetro **MAPPING-LEVEL**. Algunos parámetros no están disponibles en este nivel de tiempo de ejecución.
- 4.0** El archivo de enlace de servicios web generado se despliega correctamente en una región de CICS TS 5.2 o posterior. Con este nivel

de ejecución, puede utilizar un nivel de correlación de 4.0 o anterior para el parámetro **MAPPING-LEVEL**. Puede utilizar cualquier parámetro opcional en este nivel.

- 4.1** El archivo de enlace de servicios web generado se despliega correctamente en una región de CICS V5.2 que tiene aplicado APAR PI67641, o en cualquier región de CICS V5.3 o posterior. Con este nivel de ejecución, puede utilizar un nivel de correlación de 4.1 o anterior para el parámetro **MAPPING-LEVEL**.

CURRENT

El archivo de enlace de servicios web generado se despliega correctamente en una región de CICS al mismo nivel de ejecución que el que está utilizando para generar el archivo de enlace de servicio web.

OPERATION-NAME = *valor*

Especifica el nombre de operación que se utiliza en el documento WSDL generado. Si no se proporciona un valor, se genera un nombre predeterminado utilizando el valor del parámetro **PGMNAME** seguido del valor **operation**.

PDSLIB = *valor*

Especifica el nombre del conjunto de datos particionados que contiene las estructuras de datos de lenguaje de alto nivel que se van a procesar. Los miembros del conjunto de datos que utilizados para la solicitud y respuesta se especifican en los parámetros **REQMEM** y **RESPMEM** respectivamente.

Restricción: Los registros del conjunto de datos particionados debe tener una longitud fija de 80 bytes.

PDSCP = *valor*

Especifica la página de códigos que se utiliza en los miembros del conjunto de datos particionados especificado en los parámetros **REQMEM** y **RESPMEM**, donde *valor* es un número CCSID o un número de página de códigos de Java. Si no se especifica este parámetro, se utiliza la página de códigos de los servicios del sistema z/OS UNIX. Por ejemplo, puede especificar **PDSCP** = 037.

PGMINT = { **CHANNEL** | **COMMAREA** }

Para un proveedor de servicios, especifique cómo CICS pasa datos al programa de aplicación de destino:

CHANNEL

CICS utiliza una interfaz de canal para pasar datos al programa de aplicación de destino.

- En niveles de correlación anteriores a 3.0, el canal puede tener sólo un contenedor, que se utiliza para la entrada y la salida. Utilice el parámetro **CONTID** para especificar el nombre del contenedor. El nombre predeterminado es DFHWS-DATA.
- En el nivel de correlación 3.0, el canal puede contener varios contenedores. Utilice los parámetros **REQUEST-CHANNEL** y **RESPONSE-CHANNEL**. No especifique **PDSLIB** , **REQMEM** , o **RESPMEM**.

COMMAREA

CICS utiliza un área de comunicación (**COMMAREA**) para pasar datos al programa de aplicación de destino.

Cuando el programa de aplicación de destino ha procesado la solicitud, debe utilizar el mismo mecanismo para devolver la respuesta. Si se ha recibido la solicitud en un área de comunicación, la respuesta debe devolverse en el área de comunicación; si la solicitud se ha recibido en un contenedor, la respuesta

debe devolverse en un contenedor. La longitud del contenedor o área de comunicaciones que pasa CICS al programa de aplicación de destino es mayor que las longitudes del contenedor o área de comunicación de solicitud y el contenedor o área de comunicación de la respuesta.

PGMNAME = *valor*

Especifica el nombre del recurso PROGRAM de CICS para el programa de aplicación de destino que se expone como un servicio web. El soporte de servicio web de CICS enlazará con este programa.

PORT-NAME = *valor*

Especifica el nombre que se utiliza para el puerto y el portType en el documento WSDL generado. Si no se proporciona valor, se genera un nombre predeterminado utilizando el valor del parámetro **PGMNAME** seguido de "Port". Si SOAPVER se establece en ALL, se añade un sufijo "12" al nombre del puerto SOAP 1.2.

REQMEM = *valor*

Especifica el nombre del miembro de conjunto de datos particionados que contiene la estructura de lenguaje de alto nivel para la solicitud de servicio web. Para un proveedor de servicio, la solicitud de servicio web es la entrada al programa de aplicación.

REQUEST-CHANNEL = *valor*

Especifica el nombre y la ubicación de un documento de descripción de canal. La descripción de canal describe los contenedores que puede utilizar la aplicación de proveedor de servicios web en su interfaz cuando recibe un mensaje SOAP desde un solicitante de servicio web. La descripción de canal está en un documento XML que debe ajustarse al esquema de canal proporcionado por CICS.

Este parámetro sólo se puede utilizar en el nivel de correlación 3.0.

REQUEST-NAMESPACE = *valor*

Especifica el espacio de nombres del esquema XML para el mensaje de solicitud en la descripción de servicio web generada. Si no especifica este parámetro, CICS genera un espacio de nombres automáticamente.

RESPMEM = *valor*

Especifica el nombre del miembro de conjunto de datos particionados que contiene la estructura de lenguaje de alto nivel para la respuesta de servicio web. Para un proveedor de servicios, la respuesta de servicio web es la salida del programa de aplicación.

Omita este parámetro si no hay ninguna respuesta implicada; es decir, para mensajes unidireccionales.

RESPONSE-CHANNEL = *valor*

Especifica el nombre y la ubicación de un documento de descripción de canal. La descripción de canal describe los contenedores que puede utilizar la aplicación de proveedor de servicios web en su interfaz cuando envía un mensaje de respuesta SOAP a un solicitante de servicio web. La descripción de canal está en un documento XML que debe ajustarse al esquema de canal proporcionado por CICS.

Este parámetro sólo se puede utilizar en el nivel de correlación 3.0.

RESPONSE-NAMESPACE = *valor*

Especifica el espacio de nombres del esquema XML para el mensaje de respuesta en la descripción de servicio web generada. Si no especifica este parámetro, CICS genera un espacio de nombres automáticamente.

SERVICE-NAME = *valor*

Especifica el nombre de servicio que se utiliza en el documento WSDL generado. Si no se proporciona valor, se genera un nombre de servicio predeterminado utilizando el valor del parámetro **PGMNAME** seguido de "Service".

SOAPVER = { 1.1 | 1.2 | ALL }

Especifica el nivel SOAP a utilizar en la descripción de servicio web generada. Este parámetro está disponible sólo cuando **MINIMUM-RUNTIME-LEVEL** se establece en 2.0 o superior.

1.1 El protocolo SOAP 1.1 se utiliza como enlace para la descripción de servicio web.

1.2 El protocolo SOAP 1.2 se utiliza como enlace para la descripción de servicio web.

ALL Tanto el protocolo SOAP 1.1 como 1.2 se pueden utilizar como enlace para la descripción de servicio web.

Si no especifica un valor para este parámetro, el valor predeterminado depende de la versión de WSDL que desea crear:

- Si sólo se necesita WSDL 1.1, se utiliza el enlace de SOAP 1.1.
- Si sólo se necesita WSDL 2.0, se utiliza el enlace de SOAP 1.2.
- Si necesita WSDL 1.1 y WSDL 2.0, se utilizan ambos enlaces SOAP 1.1 y 1.2 para cada descripción de servicio web.

SSL-KEYSTORE = *valor*

Este parámetro opcional especifica la ubicación completa del archivo de almacén de claves.

Utilice este parámetro si desea que el asistente de servicios web utilice el cifrado de capa de sockets seguros (SSL) para comunicarse a través de una red con IBM WebSphere Service Registry and Repository (WSRR).

SSL-KEYPWD = *valor*

Este parámetro opcional especifica la contraseña para el almacén de claves.

Utilice este parámetro si desea que el asistente de servicios web utilice el cifrado de capa de sockets seguros (SSL) para comunicarse a través de una red con IBM WebSphere Service Registry and Repository (WSRR).

SSL-TRUSTSTORE = *valor*

Este parámetro opcional especifica la ubicación completa del archivo de almacén de confianza.

Utilice este parámetro si desea que el asistente de servicios web utilice el cifrado de capa de sockets seguros (SSL) para comunicarse a través de una red con IBM WebSphere Service Registry and Repository (WSRR).

SSL-TRUSTPWD = *valor*

Este parámetro opcional especifica la contraseña para el almacén de confianza.

Utilice este parámetro si desea que el asistente de servicios web utilice el cifrado de capa de sockets seguros (SSL) para comunicarse a través de una red con IBM WebSphere Service Registry and Repository (WSRR).

STRUCTURE = (*solicitud* , *respuesta*)

Sólo para C y C++, especifica los nombres de las estructuras de alto nivel contenidas en los miembros de conjunto de datos particionados especificados en los parámetros **REQMEM** y **RESPMEM**:

solicitud

Especifica el nombre de la estructura de alto nivel que contiene la solicitud cuando se especifica el parámetro **REQMEM**. El valor predeterminado es DFHREQUEST.

El miembro de conjunto de datos particionados debe contener una estructura de alto nivel con el nombre que especifica o una estructura denominada DFHREQUEST si no especifica un nombre.

respuesta

Especifica el nombre de la estructura de alto nivel que contiene la respuesta cuando se especifica el parámetro **RESPMEM**. El valor predeterminado es DFHRESPONSE.

Si especifica un valor, el miembro de conjunto de datos particionados debe contener una estructura de alto nivel con el nombre que especifica o una estructura denominada DFHRESPONSE si no especifica un nombre.

SYNCONRETURN = { **NO** | **YES** }

Especifica si el servicio web remoto puede emitir un punto de sincronización.

NO El servicio web remoto no puede emitir un punto de sincronización. Este valor es el predeterminado. Si el servicio web remoto emite un punto de sincronización, falla con una terminación anómala ADPL.

YES El servicio web remoto puede emitir un punto de sincronización. Si selecciona YES, la tarea remota se confirma como una unidad de trabajo individual cuando se devuelve el control desde el servicio web remoto. Si el servicio web remoto actualiza un recurso recuperable y se produce un fallo después de que se devuelve, la actualización para ese recurso no se puede restituir.

TRANSACTION = *nombre*

En un proveedor de servicios, este parámetro especifica de uno a cuatro caracteres del nombre de una transacción de alias que puede iniciar la interconexión. El valor de este parámetro se utiliza para definir el atributo TRANSACTION del recurso URIMAP cuando se crea automáticamente utilizando el mandato de exploración **PIPELINE**.

Caracteres aceptables:

A-Z a-z 0-9 \$ @ # _ < >

TRUNCATE-NULL-ARRAYS = { **DISABLED** | **ENABLED** }

Especifica cómo se procesan las matrices estructuradas en el nivel de correlación 4.1 o superior. Si está habilitado, CICS intentará reconocer los registros vacíos dentro de una matriz (consulte TRUNCATE-NULL-ARRAY-VALUES para obtener más información sobre la identificación de registros vacíos). Si se detectan cinco registros de matriz vacíos consecutivos, la matriz se trunca en el primero de esos registros cuando se genera XML/JSON. Esta prestación de recorte sólo se habilita para matrices con contenido estructurado, las matrices de campos primitivos simples no están sujetos al recorte. El recorte de matrices puede dar como resultado una representación más precisa de los datos en JSON/XML, pero no sin riesgo. Si cinco registros de datos consecutivos se identifican de forma incorrecta como almacenamiento no inicializado (quizás porque contienen legítimamente valores bajos), se puede experimentar una pérdida de datos. Si TRUNCATE-NULL-ARRAYS está habilitado y TRUNCATE-NULL-ARRAY-VALUES no se ha establecido, se utiliza el valor predeterminado para TRUNCATE-NULL-ARRAY-VALUES.

TRUNCATE-NULL-ARRAY-VALUES = { NULL | **SPACE** | **ZERO** }

Especifica qué valores se tratan como vacíos para el proceso de TRUNCATE-NULL-ARRAYS en el nivel de correlación 4.1 o superior. De forma predeterminada, el valor nulo (0x00, o los valores bajos) se tratan como vacíos. Si todos los bytes de almacenamiento dentro de un registro de una matriz estructurada contienen nulos, entonces todo el registro se considera vacío. Se pueden especificar uno o más de los valores NULL, SPACE y ZERO en una lista separada por comas, donde NULL implica un carácter nulo (0x00), SPACE implica un SBCS EBCDIC Space (0x40) y ZERO implica un cero decimal con zona (0xF0). Cualquier combinación coincidente de los bytes seleccionados dentro un registro de matriz estructurado provocará que todo el registro se identifique como vacío. Si TRUNCATE-NULL-ARRAY-VALUES tiene un valor definido, entonces TRUNCATE-NULL-ARRAYS debe estar habilitado.

URI = *valor*

Este parámetro especifica el URI absoluto o relativo que utilizará un cliente para acceder al servicio web. CICS utiliza el valor que se especifica cuando genera un recurso URIMAP desde un archivo de enlace de servicio web que ha creado DFHLS2WS. El parámetro especifica el componente de vía de acceso del URI al que se aplica la definición URIMAP.

USERID = *id*

En un proveedor de servicios, este parámetro especifica un ID de usuario de 1 a 8 caracteres, que puede utilizar cualquier cliente web. Para una respuesta generada por la aplicación o un servicio web, la transacción de alias se conecta bajo este ID de usuario. El valor de este parámetro se utiliza para definir el atributo USERID del recurso URIMAP cuando se crea automáticamente utilizando el mandato de exploración **PIPELINE**.

Caracteres aceptables:

A-Z a-z 0-9 \$ @ #

WSBIND = *valor*

El nombre completo del archivo de enlace de servicio web de z/OS UNIX. DFHLS2WS crea el archivo, pero no la estructura de directorio, si no existe. La extensión de archivo es .wsbind.

WSDL = *valor*

El nombre completo del archivo de z/OS UNIX donde se escribe la descripción de servicio web. La descripción de servicio web no se ajusta a la especificación WSDL 1.1. DFHLS2WS crea el archivo, pero no la estructura de directorio, si no existe. La extensión de archivo es .wsdl.

WSDL_1.1 = *valor*

El nombre completo del archivo de z/OS UNIX donde se escribe la descripción de servicio web. La descripción de servicio web no se ajusta a la especificación WSDL 1.1. DFHLS2WS crea el archivo, pero no la estructura de directorio, si no existe. La extensión de archivo es .wsdl. Este parámetro da el mismo resultado que el parámetro **WSDL**, por lo tanto puede especificar sólo uno o el otro.

WSDL_2.0 = *valor*

El nombre completo del archivo de z/OS UNIX donde se escribe la descripción de servicio web. La descripción de servicio web no se ajusta a la especificación WSDL 2.0. DFHLS2WS crea el archivo, pero no la estructura de directorio, si no existe. La extensión de archivo es .wsdl. Este parámetro se puede utilizar con los parámetros **WSDL** o **WSDL_1.1**. Está disponible sólo cuando **MINIMUM-RUNTIME-LEVEL** se establece en 2.0 o superior.

WSDLCP = { **LOCAL** | **UTF-8** | **EBCDIC-CP-US** }

Especifica la página de códigos que se utiliza para generar el documento WSDL.

LOCAL

Especifica que el documento WSDL se genera utilizando la página de códigos local y no se genera ninguna etiqueta de codificación en el documento WSDL.

UTF-8 Especifica que el documento WSDL se genera utilizando la página de códigos UTF-8. Se genera una etiqueta de codificación en el documento WSDL. Si especifica esta opción, debe asegurarse de que la codificación permanece correcta cuando copia el documento WSDL entre diferentes plataformas.

EBCDIC-CP-US

Este valor especifica que el documento WSDL se genera utilizando la página de códigos US EBCDIC. Se genera una etiqueta de codificación en el documento WSDL.

WSDL-NAMESPACE = *valor*

Especifica el espacio de nombres para CICS para utilizarlo en el documento WSDL generado.

Si no especifica este parámetro, CICS genera un espacio de nombres automáticamente.

WSRR-CUSTOM- *PropertyName* = *valor*

Utilice este parámetro opcional para añadir metadatos personalizados al documento WSDL en el WSRR. Los pares **WSRR-CUSTOM- *PropertyName*** = *valor* se añaden al documento WSDL y aparecen en WSRR sin el prefijo **WSRR-CUSTOM**.

Puede especificar un máximo de 255 pares *PropertyName* = *valor* personalizados. Evite duplicar y dejar en blanco los pares *PropertyName* = *valor*.

Utilice este parámetro sólo cuando se especifique el parámetro **WSRR-SERVER**.

WSRR-DESCRIPTION = *valor*

Utilice este parámetro opcional para especificar los metadatos que describe el documento WSDL que se está publicando.

Utilice este parámetro sólo cuando se especifique el parámetro **WSRR-SERVER**.

WSRR-ENCODING = *valor*

Utilice este parámetro opcional para especificar la codificación del conjunto de caracteres del documento WSDL. Si no se especifica el parámetro **WSRR-ENCODING**, WSRR utiliza el valor especifica en el documento WSDL.

Utilice este parámetro sólo cuando se especifique el parámetro **WSRR-SERVER**.

WSRR-LOCATION = *valor*

Utilice este parámetro opcional para especificar el URI que identifica la ubicación del documento WSDL. Si este parámetro no se especifica, el URI utiliza de forma predeterminada el nombre de archivo del parámetro **WSDL**. Por ejemplo, si el valor del parámetro **WSDL** es `wsrr/example.wsdl`, el valor del parámetro **WSRR-LOCATION** utiliza el valor predeterminado de `example.wsdl`.

Utilice este parámetro sólo cuando se especifique el parámetro **WSRR-SERVER**.

WSRR-PASSWORD = *valor*

Utilice este parámetro opcional si debe entrar una contraseña para acceder a WSRR.

Si se especifica el parámetro **WSRR-USERNAME**, también debe especificar este parámetro.

Utilice este parámetro sólo cuando se especifique el parámetro **WSRR-SERVER**.

WSRR-SERVER = { *nombre de dominio : número de puerto | dirección IP : número de puerto* }

Utilice este parámetro para especificar la ubicación del servidor IBM WebSphere Service Registry and Repository (WSRR). Si se especifica este parámetro, se utiliza la validación del parámetro WSRR.

WSRR-USERNAME = *valor*

Utilice este parámetro opcional si necesita especificar un nombre de usuario para acceder a WSRR. WSRR utiliza este nombre de usuario para establecer la propiedad de propietario.

Utilice este parámetro sólo cuando se especifique el parámetro **WSRR-SERVER**.

WSRR-VERSION = { 1 | *valor* }

Utilice este parámetro para establecer la propiedad de la versión del documento WSDL en WSRR.

Utilice este parámetro sólo cuando se especifique el parámetro **WSRR-SERVER**.

Otra información

- El ID de usuario bajo el que se ejecuta DFHLS2SC debe configurarse para utilizar los servicios de sistema UNIX. El ID de usuario debe tener permiso de lectura para la estructura de archivo de CICS z/OS UNIX y las bibliotecas PDS y permiso de escritura para los directorio especificados en los parámetros **LOGFILE**, **WSBIND** y **WSDL**.
- El ID de usuario debe tener una asignación de almacenamiento lo suficientemente grande para ejecutar Java.
- El JCL tiene una longitud de parámetro máxima de 100 caracteres. Esto se puede aumentar utilizando la sentencia **STDPARM**; para obtener más información, consulte *z/OS UNIX System Services User Guide*.

Ejemplo

```
//LS2WS JOB '  
accounting information  
'  
name,MSGCLASS=A  
// SET QT='''  
//JAVAPROG EXEC DFHLS2WS,  
// TMPFILE=&QT.&SYSUID.&QT  
//INPUT.SYSUT1 DD *  
PDSLIB=//CICSHLQ.SDFHSAMP  
REQMEM=DFH0XCP4  
RESPMEM=DFH0XCP4  
LANG=COBOL  
LOGFILE=/u/exampleapp/wsbinding/example.log  
MINIMUM-RUNTIME-LEVEL=2.1  
MAPPING-LEVEL=2.1  
CHAR-VARYING=COLLAPSE  
PGMNAME=DFH0XCMN  
URI=http://myserver.example.org:8080/exampleApp/example  
PGMINT=COMMAREA  
SOAPVER=1.1  
SYNCONRETURN=YES  
WSBIND=/u/exampleapp/wsbinding/example.wsbinding  
WSDL=/u/exampleapp/wsd1/example.wsd1
```

```

WSDL_2.0=/u/exampleapp/wsd1/example_20.wsd1
WSDLCP=LOCAL
WSDL-NAMESPACE=http://mywsdlnamespace
/*

```

DFHWS2LS: Conversión de WSDL a lenguaje de alto nivel

El procedimiento DFHWS2LS genera una estructura de datos de lenguaje de alto nivel y un archivo de enlace de servicio web desde una descripción de servicio web. Puede utilizar DFHWS2LS cuando expone un programa de aplicación CICS como proveedor de servicios o cuando construye un solicitante de servicio.

Sentencias de control de trabajos para DFHWS2LS

JOB Inicia el trabajo.

EXEC Especifica el nombre de procedimiento (DFHWS2LS).

INPUT.SYSUT1 DD

Especifica la entrada. Los parámetros de entrada se especifican normalmente en el flujo de entrada. Sin embargo, se pueden definir en un conjunto de datos en miembro de un conjunto de datos particionados.

Parámetros simbólicos

Los siguientes parámetros simbólicos se definen en DFHWS2LS:

JAVADIR = *vía de acceso*

Especifica el nombre del directorio Java que utiliza DFHWS2LS. El valor de este parámetro se añade a /usr/lpp/ para crear un nombre de vía de acceso completo de /usr/lpp/ *vía de acceso* .

Generalmente, no es necesario especificar este parámetro. El valor predeterminado es el valor proporcionado para el trabajo de instalación (DFHISTAR) de CICS en el parámetro **JAVADIR**.

PATHPREFIX = *prefijo*

Especifica un prefijo opcional que amplía la vía de acceso del directorio z/OS UNIX utilizado en otros parámetros. El valor predeterminado es la serie vacía.

Generalmente, no es necesario especificar este parámetro. El valor predeterminado es el valor proporcionado para el trabajo de instalación (DFHISTAR) de CICS en el parámetro **PATHPREFIX**.

TMPDIR = *tmpdir*

Especifica la ubicación de un directorio en z/OS UNIX que DFHWS2LS utiliza como espacio de trabajo temporal. El ID de usuario bajo el que se ejecuta el trabajo debe tener permiso de lectura y escritura para este directorio.

El valor predeterminado es /tmp.

TMPPFILE = *tmpprefix*

Especifica un prefijo que DFHWS2LS utiliza para construir los nombres de los archivos de espacio de trabajo temporal.

El valor predeterminado es WS2LS.

USSDIR = *vía de acceso*

Especifica el nombre del directorio CICS TS en el sistema de archivos de servicios del sistema UNIX. El valor de este parámetro se añade a /usr/lpp/cicsts/ para crear un nombre de vía de acceso completa de /usr/lpp/cicsts/ *path* .

Generalmente, no es necesario especificar este parámetro. El valor predeterminado es el valor proporcionado para el trabajo de instalación (DFHISTAR) de CICS en el parámetro **USSDIR**.

SERVICE = *valor*

Utilice este parámetro sólo cuando el soporte de IBM le indique que lo haga.

Espacio de trabajo temporal

DFHWS2LS crea estos tres archivos temporales en tiempo de ejecución:

```
tmpdir / tmpprefix .in
tmpdir / tmpprefix .out
tmpdir / tmpprefix .err
```

donde:

tmpdir es el valor que se especifica en el parámetro **TMPDIR**.

tmpprefix es el valor que se especifica en el parámetro **TMPFILE**.

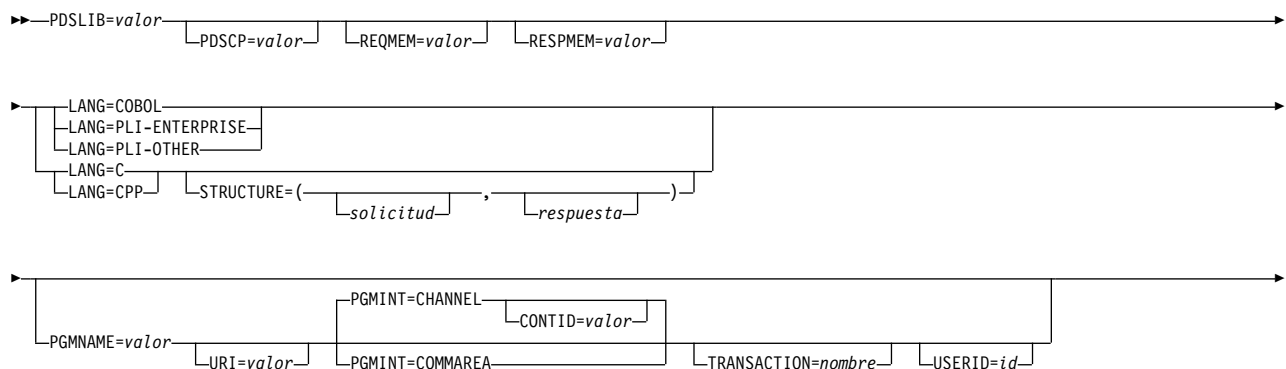
Los nombres predeterminados para los archivos, cuando no se especifica **TMPDIR** y **TMPFILE** son los siguientes:

```
/tmp/WS2LS.in
/tmp/WS2LS.out
/tmp/WS2LS.err
```

Importante: DFHWS2LS no bloquea el acceso a los archivos z/OS UNIX o los miembros del conjunto de datos. Por lo tanto, si dos o más instancias de DFHWS2LS se ejecutan de forma simultánea y utilizan los mismos archivos de espacio de trabajo temporal, nada impide a un trabajo sobrescribir los archivos de espacio de trabajo mientras otro trabajo los está utilizando, llevando a errores imprevisibles.

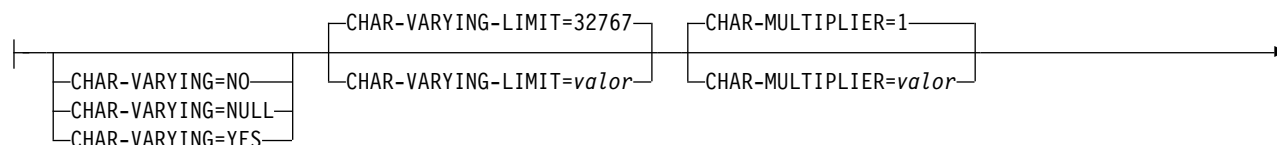
Por lo tanto, se aconseja crear un convenio de denominación y procedimientos operativos que eviten esta situación. Por ejemplo, puede utilizar el parámetro simbólico de sistema **SYSUID** para generar nombres de archivo de espacios de trabajo que son exclusivos para un usuario individual. Estos archivos temporales se suprimen antes de la finalización del trabajo.

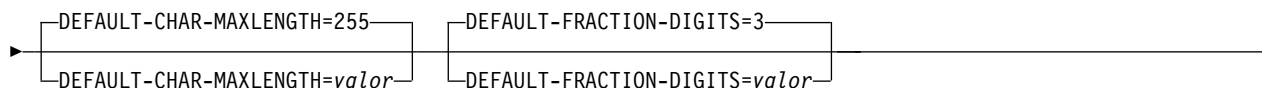
Parámetros de entrada de DFHWS2LS



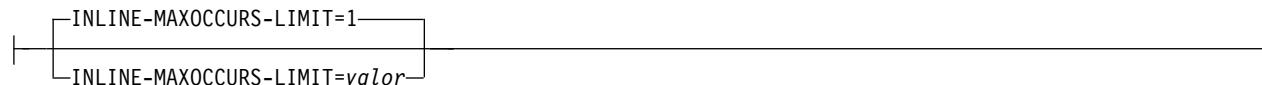


Correlación de datos avanzada (nivel de correlación 1.2 y superior):

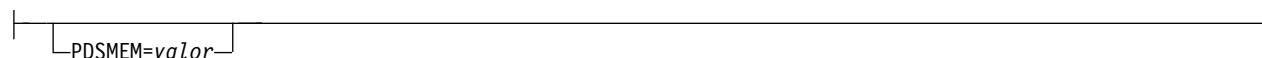




Correlación de datos avanzada (nivel de correlación 2.1 y superior):



Correlación de datos avanzada (nivel de correlación 2.2 y superior):



Correlación de datos avanzada (nivel de tiempo de ejecución 2.1 y superior):



Correlación de datos avanzada (nivel de tiempo de ejecución 3.0 y superior):



Notas:

- 1 Cada uno de los parámetros WSRR que se pueden especificar cuando se establece el parámetro **WSRR-SERVER** sólo se pueden especificar una vez.

Utilización de parámetros

- Puede especificar los parámetros de entrada en cualquier orden.
- Cada parámetro se debe iniciar en una nueva línea.
- Un parámetro, y su carácter de continuación, si utiliza uno, no deben sobrepasar la columna 72; las columnas 73 a 80 deben contener espacios en blanco.
- Si un parámetro es demasiado largo para ajustarse en una única línea, utilice el carácter de asterisco (*) al final de la línea para indicar que el parámetro continúa en la línea siguiente. Todo, incluidos los espacios antes del asterisco, se considera parte del parámetro. Por ejemplo:

```
WSBIND=wsbinddir*
      /app1
```

es equivalente a

```
WSBIND=wsbinddir/app1
```

- Un carácter # en la primera posición de carácter de la línea es un carácter de comentario. La línea se ignora.
- Una coma en la última posición de carácter de la línea es un separador de línea opcional y se ignora.

Descripciones de parámetros

BINDING = *valor*

Si la descripción de servicio web contiene más de un elemento `<wsdl:Binding>`, utilice este parámetro para especificar cuál se utiliza para generar la estructura de lenguaje y el archivo de enlace de servicio web. Especifique el valor del atributo `name` que se utiliza en el elemento `<wsdl:Binding>` en la descripción de servicio web.

CCSID = *valor*

Especifica el CCSID que se utiliza en tiempo de ejecución para codificar datos de caracteres en la estructura de datos de la aplicación. El valor de este parámetro sustituye el valor del parámetro de inicialización del sistema **LOCALCCSID**. El *valor* debe ser un EBCDIC CCSID soportado por los servicios de conversión Java y z/OS (consulte z/OS Unicode Services User's Guide and Reference). Si no especifica este parámetro, la estructura de datos de aplicación se codifica utilizando el CCSID especificado en el parámetro de inicialización del sistema.

Puede utilizar este parámetro con cualquier nivel de correlación.

CHAR-MULTIPLIER = { 1 | *valor* }

Especifica el número de bytes que se permiten para cada carácter cuando el nivel de correlación es 1.2 o posterior. El *valor* de este parámetro puede ser un entero positivo en el rango de 1 a 2.147.483.647. Todas las correlaciones basadas en caracteres no numéricos están sujetas a este multiplicador. Los campos binario, numérico, con zona y decimal empaquetado no están sujetos a este multiplicador.

Este parámetro puede ser útil si, por ejemplo, está planificando utilizar caracteres DBCS donde puede optar por un multiplicador de 3 para permitir espacio para caracteres de cambio y desplazamiento potenciales alrededor de todos los caracteres de doble byte en tiempo de ejecución.

Cuando establece **CCSID=1200** (indicando UTF-16), los únicos valores válidos para **CHAR-MULTIPLIER** son 2 o 4 . Cuando utiliza UTF-16, el valor predeterminado es 2 . Utilice **CHAR-MULTIPLIER=2** cuando espera que los datos de aplicación contengan caracteres que requieren 1 unidad de codificación UTF-16. Utilice **CHAR-MULTIPLIER=4** cuando espera que los datos de aplicación contengan caracteres que requieren 2 unidades de codificación UTF-16.

Nota: Establecer **CHAR-MULTIPLIER** en 1 no excluye el uso de caracteres DBCS, y establecerlo en 2 no excluye el uso de pares de sustitución UTF-16. Sin embargo, si los caracteres amplios utilizan de forma rutinaria algunos valores válidos no se ajustarán al campo asignado. Si se utiliza un valor **CHAR-MULTIPLIER** mayor, es posible almacenar más caracteres en el campo asignado que los que son válidos en el XML. Se debe tener cuidado para ajustarse a las restricciones de rango adecuadas.

CHAR-VARYING = { **NO** | **NULL** | **YES** }

Especifica cómo se correlacionan datos de caracteres de longitud variable cuando el nivel de correlación es 1.2 o superior. Los tipos de datos binarios de longitud variable se correlacionan siempre con un contenedor o una estructura variable. Si no especifica este parámetro, la correlación predeterminada depende del lenguaje especificado. Puede seleccionar estas opciones:

NO Los datos de caracteres de longitud variable se correlacionan como series de longitud fija.

NULL Los datos de caracteres de longitud variable se correlacionan con series terminadas en nulo.

YES Los datos de caracteres de longitud variable se correlacionan con un tipo de datos CHAR VARYING en PL/I. En los lenguajes COBOL, C y C++, los datos de caracteres de longitud variable se correlacionan con una representación equivalente que consta de dos elementos relacionados: la longitud de datos y los datos.

CHAR-VARYING-LIMIT = { 32767 | *valor* }

Especifica el tamaño máximo de datos binarios y de datos de caracteres de longitud variable que se correlacionan con la estructura de lenguaje cuando el nivel de correlación es 1.2 o superior. Si los datos binarios o de caracteres son mayores que el valor especificado en este parámetro, se correlacionan con un contenedor y el nombre de contenedor se utiliza en la estructura de lenguaje generada. El valor puede estar comprendido entre 0 y el valor predeterminado 32.767 bytes.

CONTID = *valor*

En un proveedor de servicios, especifique el nombre del contenedor que tiene la estructura de datos de nivel superior utilizada para representar un mensaje SOAP.

La longitud del contenedor que CICS pasa al programa de aplicación de destino es mayor que las longitudes del contenedor de solicitud y el contenedor de respuesta.

DATA-SCREENING = { ENABLED | **DISABLED** }

Especifica si los datos proporcionados por la aplicación se analizan para detectar errores.

ENABLED

Los datos de tiempo de ejecución proporcionados por la aplicación que son incoherentes con la estructura de lenguaje, se tratan como un error y se emite un mensaje DFHPI1010. Se devuelve una respuesta de error a la aplicación.

DISABLED

Los valores de datos de los datos de tiempo de ejecución proporcionados por la aplicación que son incoherentes con la estructura de lenguaje se sustituyen por valores predeterminados. Por ejemplo, un cero sustituye un valor erróneo en un campo numérico. No se emite mensaje DFHPI1010 y se devuelve una respuesta normal a la aplicación. Esta característica se puede utilizar para evitar las respuestas de error INVALID_PACKED_DEC y INVALID_ZONED_DEC generadas desde los campos de salida no inicializados.

DATA-TRUNCATION = { DISABLED | **ENABLED** }

Especifica si los datos de longitud variable se soportan en una estructura de campo de longitud fija:

DISABLED

Si los datos son menores que la longitud fija que está esperando CICS, CICS rechaza los datos truncados y emite un mensaje de error.

ENABLED

Si los datos son menores que la longitud fija que está esperando CICS, CICS tolera los datos truncados y procesa los datos que faltan como valores nulos

DATETIME = { **PACKED15** | **STRING** }

Especifica cómo se correlacionan los elementos <xsd:dateTime> con la estructura de lenguaje.

PACKED15

El valor predeterminada es que cualquier elemento <xsd:dateTime> se procesa como una indicación de fecha y hora y se correlaciona con el formato ABSTIME de CICS.

STRING

El elemento <xsd:dateTime> se procesa como texto.

DEFAULT-CHAR-MAXLENGTH = { **255** | *valor* }

Especifica la longitud de matriz predeterminada de los datos de caracteres en caracteres para correlaciones donde no está implicada la longitud en el documento de descripción servicio web, cuando el nivel de correlación es 1.2 o posterior. El valor de este parámetro puede ser un entero positivo en el rango de 1 a 2.147.483.647.

DEFAULT-FRACTION-DIGITS = { **3** | *valor* }

Especifica el número de dígitos de fracción predeterminado a utilizar en un tipo de esquema decimal XML. El valor predeterminado es 3. Para COBOL, el rango válido es 0-17, o 0-30 si se está utilizando **WIDE-COMP3**. Para C o PLI, el rango válido es 0-30.

HTTPPROXY = { *nombre de dominio : número de puerto* | *dirección IP : número de puerto* }

Si el WSDL contiene referencias a otros archivos WSDL ubicados en Internet, y el sistema en el que está ejecutando DFHWS2LS utiliza un servidor proxy para acceder a Internet, especifique el nombre de dominio o dirección IP y el número de puerto del servidor proxy. Por ejemplo:

HTTPPROXY=proxy.example.com:8080

En otros casos, este parámetro no es necesario.

HTTPPROXY-PASSWORD = *valor*

Especifica la contraseña del proxy HTTP que se debe utilizar con **HTTPPROXY-USERNAME** si el sistema en el que se está ejecutando DFHWS2LS utiliza un servidor proxy HTTP para acceder a Internet, y el servidor proxy HTTP utiliza autenticación básica. Puede utilizar este parámetro sólo cuando especifica **HTTPPROXY**.

HTTPPROXY-USERNAME = *valor*

Especifica el nombre de usuario del proxy HTTP que se debe utilizar con **HTTPPROXY-PASSWORD** si el sistema en el que se está ejecutando DFHWS2LS utiliza un servidor proxy HTTP para acceder a Internet, y el servidor proxy HTTP utiliza autenticación básica. Puede utilizar este parámetro sólo cuando especifica **HTTPPROXY**.

INLINE-MAXOCCURS-LIMIT = { **1** | *valor* }

Especifica si se utiliza el contenido de repetición variable en línea basado en el atributo maxOccurs. El contenido de repetición variable que se correlaciona en línea se coloca en el contenedor actual con el resto de la estructura de lenguaje generada. El contenido de repetición variable se almacena en dos partes, como contador que almacena el número de apariciones de los datos y como una matriz que almacena cada aparición de los datos. La correlación alternativa para el contenido de repetición variable es una correlación basada en contenido, que almacena el número de apariciones de los datos y el nombre

del contenedor donde se colocan los datos. Almacenar los datos en un contenedor distinto tiene implicaciones de rendimiento que pueden hacer que sea preferible la correlación en línea.

El parámetro **INLINE-MAXOCCURS-LIMIT** está disponible sólo del nivel de correlación 2.1 en adelante. El valor de **INLINE-MAXOCCURS-LIMIT** puede ser un entero positivo en el rango de 0 a 32.767. Un valor de 0 indica que no se utiliza la correlación en línea. Un valor de 1 garantiza que los elementos opcionales se correlacionan en línea. Si el *valor* del atributo `maxOccurs` es mayor del *valor* de **INLINE-MAXOCCURS-LIMIT**, se utiliza la correlación basada en contenedor; de lo contrario, se utiliza la correlación en línea.

Cuando decide si desea que las listas de repetición variable se correlacionen en línea, tenga en cuenta la longitud de un único elemento de datos recurrentes. Si se producen muchas instancias de longitud larga, es preferible la correlación basada en contenedor; si se producen muchas instancias de longitud corta, es preferible la correlación en línea.

LANG = COBOL

Especifica que el lenguaje de programación de la estructura de lenguaje de alto nivel es COBOL.

LANG = PLI-ENTERPRISE

Especifica que el lenguaje de programación de la estructura de lenguaje de alto nivel es Enterprise PL/I.

LANG = PLI-OTHER

Especifica que el lenguaje de programación de la estructura de lenguaje de alto nivel es un nivel de PL/I distinto a Enterprise PL/I.

LANG = C

Especifica que el lenguaje de programación de la estructura de lenguaje de alto nivel es C.

LANG = CPP

Especifica que el lenguaje de programación de la estructura de lenguaje de alto nivel es C++.

LOGFILE = *valor*

El nombre completo del archivo de z/OS UNIX en el que DFHWS2LS escribe la información de rastreo y registro de actividad. DFHWS2LS crea el archivo, pero no la estructura de directorio, si no existe.

Normalmente, no utiliza este archivo, pero puede solicitarlo la organización de servicio de IBM si encuentra problemas con DFHWS2LS.

MAPPING-LEVEL = { 1.0 | 1.1 | 1.2 | 2.0 | 2.1 | 2.2 | 3.0 | 4.0 | 4.1 }

Especifica el nivel de correlación que utiliza DFHWS2LS cuando genera el archivo de enlace de servicio web y la estructura de lenguaje. Puede seleccionar estas opciones:

- 1.0** El archivo de enlace de servicio web y la estructura de lenguaje se generan utilizando niveles de correlación CICS TS 3.1.
- 1.1** Los atributos XML y los tipos de datos `<list>` y `<union>` se correlacionan con la estructura de lenguaje. Los datos binarios y de caracteres que tienen una longitud máxima de más de 32.767 se correlacionan con un contenedor. El nombre de contenedor se crea en la estructura de lenguaje.
- 1.2** Utilice los parámetros **CHAR-VARYING** y **CHAR-VARYING-LIMIT** para controlar cómo se correlacionan y procesan los datos de caracteres en

tiempo de ejecución. Si no especifica cualquiera de estos parámetros, los datos de caracteres o binarios que tienen una longitud máxima de menos de 32.768 bytes se correlacionan con una estructura VARYING para todos los lenguajes excepto C++, donde los datos de caracteres se correlacionan con una serie terminada en nulo.

- 2.0 Utilice este nivel de correlación en una región de CICS TS 3.2 o posterior para aprovecharse de las mejoras en las correlaciones entre la estructura de lenguaje y el archivo de enlace de servicios web.
- 2.1 Utilice este nivel de correlación con una región de CICS TS 3.2 que tiene aplicado APAR PK59794, o cualquier región posterior a CICS TS 3.2 para el soporte <xsd:any> y xsd:anyType, la opción para correlacionar contenido de repetición variables en línea con el parámetro **INLINE-MAXOCCURS-LIMIT**, y soporte para minOccurs="0" en <xsd:sequence> , <xsd:choice> y <xsd:all>.
- 2.2 Utilice este nivel de correlación con una región de CICS TS 3.2 que tiene aplicado APAR PK69738 o con cualquier región posterior a CICS TS 3.2. Proporciona el siguiente soporte:
 - Elementos con valores fijos
 - Soporte mejorado para elementos <xsd:choice>
 - Tipos de datos abstractos
 - Elementos abstractos
 - Grupos de sustitución
- 3.0 Utilice este nivel de correlación con una región de CICS TS 4.1 o región posterior. En este nivel de correlación, puede transformar las indicaciones de fecha y hora en formato ABSTIME de CICS.
- 4.0 Utilice este nivel de correlación con una región de CICS TS 5.2 o posterior cuando desee utilizar UTF-16.
- 4.1 Para un soporte de matriz truncable, utilice este nivel de correlación con una región de CICS TS V5.2 que tiene aplicado APAR PI67641, o cualquier región de CICS TS V5.3 o posterior.

Para obtener más información sobre los niveles de correlación, consulte Mapping levels for the CICS assistants.

MAPPING-OVERRIDES = { SAME-AS-MAPPING-LEVEL | HYPHENS-AS-UNDERSCORES | LESS-DUP-NAMES | NO-ARRAY-NAME-INDEXING | UNDERSCORES-AS-HYPHENS }

Especifica si el comportamiento predeterminado se sustituye para el nivel de correlación especificado cuando se generan estructuras de lenguaje.

Nota: Se puede utilizar cualquiera de las subopciones en una lista delimitada por comas. Las opciones no son mutuamente excluyentes; son combinatorias y no están ordenadas.

SAME-AS-MAPPING-LEVEL

Este parámetro genera estructuras de lenguaje en el mismo estilo que el nivel de correlación. Es el valor predeterminado.

HYPHENS-AS-UNDERSCORES

Sólo para PL/I. Este parámetro convierte cualquier guión del documento WSDL en guiones bajos en lugar del carácter X, para mejorar la legibilidad de las estructuras de lenguaje PL/I generadas. Para obtener más información, consulte el apartado XML schema to PL/I mapping.

LESS-DUP-NAMES

Este parámetro genera nombres de campo de estructura no estructural con `_value` al final del nombre para habilitar la referencia directa al campo. Por ejemplo, en la siguiente estructura PLI, cuando se especifica `MAPPING-OVERRIDES=LESS-DUP-NAMES`, el campo de nivel 12 `streetName` lleva el sufijo `_value` :

```
09 streetName,  
12 streetName CHAR(255) VARYING  
UNALIGNED,  
12 filler BIT (7),  
12 attr_nil_streetName_value BIT (1),
```

La estructura resultante es la siguiente:

```
09 streetName,  
12 streetName_value CHAR(255) VARYING  
UNALIGNED,  
12 filler BIT (7),  
12 attr_nil_streetName_value BIT (1),
```

NO-ARRAY-NAME-INDEXING

Sólo para Enterprise PL/I. Garantiza que los nombres de campo dentro de una matriz sean exclusivos sólo dentro del ámbito de la estructura de nivel superior.

UNDERScores-AS-HYPHENS

Esta opción está habilitada automáticamente en el nivel de correlación 4.0.

Solo para COBOL. Este parámetro convierte cualquier guión bajo en el documento WSDL en guiones, en lugar del carácter X, para mejorar la legibilidad de las estructuras de lenguaje COBOL generadas. Si se produce un conflicto de nombres de campo, los campos se numeran para garantizar que con exclusivos. Para obtener más información, consulte el apartado XML schema to COBOL mapping.

MINIMUM-RUNTIME-LEVEL = { MINIMUM | 1.0 | 1.1 | 1.2 | 2.0 | 2.1 | 2.2 | 3.0
| 4.0 | 4.1 | **CURRENT** }

Especifica el entorno de ejecución de CICS mínimo en el que se puede desplegar el archivo de enlace de servicio web. Si selecciona un nivel que no coincide con los demás parámetros que ha especificado, recibe un mensaje de error. Puede seleccionar estas opciones:

MINIMUM

El nivel de tiempo de ejecución más bajo posible de CICS se asigna automáticamente a los parámetros determinados que seleccione.

- 1.0** El archivo de enlace de servicios web generado se despliega correctamente en una región de CICS TS 3.1 que no tiene aplicado APARs PK15904 y PK23547. Algunos parámetros no están disponibles en este nivel de tiempo de ejecución.
- 1.1** El archivo de enlace de servicios web generado se despliega correctamente en una región de CICS TS 3.1 que tiene aplicado al menos APAR PK15904. Puede utilizar un nivel de correlación de 1.1 o anterior para el parámetro `MAPPING-LEVEL`. Algunos parámetros no están disponibles en este nivel de tiempo de ejecución.
- 1.2** El archivo de enlace de servicios web generado se despliega correctamente en una región de CICS TS 3.1 que tiene aplicado APAR PK15904 y PK23547. Puede utilizar un nivel de correlación de 1.2 o

anterior para el parámetro MAPPING-LEVEL. Algunos parámetros no están disponibles en este nivel de tiempo de ejecución.

- 2.0 El archivo de enlace de servicios web generado se despliega correctamente en una región de CICS TS 3.2 o posterior. Puede utilizar un nivel de correlación de 2.0 o anterior para el parámetro MAPPING-LEVEL. Algunos parámetros no están disponibles en este nivel de tiempo de ejecución.
- 2.1 El archivo de enlace de servicios web generado se despliega correctamente en una región de CICS TS 3.2 que tiene aplicado APAR PK59794, o en cualquier región posterior a CICS TS 3.2. Puede utilizar un nivel de correlación de 2.1 o anterior para el parámetro **MAPPING-LEVEL**. Algunos parámetros no están disponibles en este nivel de tiempo de ejecución.
- 2.2 El archivo de enlace de servicios web generado se despliega correctamente en una región de CICS TS 3.2 que tiene aplicado APAR PK69738, o en cualquier región posterior a CICS TS 3.2. Con este nivel de ejecución, puede utilizar un nivel de correlación de 2.2 o anterior para el parámetro **MAPPING-LEVEL**. Algunos parámetros no están disponibles en este nivel de tiempo de ejecución.
- 3.0 El archivo de enlace de servicios web generado se despliega correctamente en una región de CICS TS 4.1 o posterior. Con este nivel de ejecución, puede utilizar un nivel de correlación de 3.0 o anterior para el parámetro **MAPPING-LEVEL**. Algunos parámetros no están disponibles en este nivel de tiempo de ejecución.
- 4.0 El archivo de enlace de servicios web generado se despliega correctamente en una región de CICS TS 5.2 o posterior. Con este nivel de ejecución, puede utilizar un nivel de correlación de 4.0 o anterior para el parámetro **MAPPING-LEVEL**. Puede utilizar cualquier parámetro opcional en este nivel.
- 4.1 El archivo de enlace de servicios web generado se despliega correctamente en una región de CICS V5.2 que tiene aplicado APAR PI67641, o en cualquier región de CICS V5.3 o posterior. Con este nivel de ejecución, puede utilizar un nivel de correlación de 4.1 o anterior para el parámetro MAPPING-LEVEL.

CURRENT

El archivo de enlace de servicios web generado se despliega correctamente en una región de CICS al mismo nivel de ejecución que el que está utilizando para generar el archivo de enlace de servicio web.

NAME-TRUNCATION = { LEFT | RIGHT }

Especifica si los nombres del elemento XML están trucados desde la izquierda o la derecha. El asistente de servicios web de CICS trunca nombres del elemento XML en la longitud adecuada para el lenguaje de alto nivel especificado; de forma predeterminada los nombres se truncan desde la derecha.

OPERATIONS = *valor*

Para las aplicaciones de solicitante de servicio web, especifique un subconjunto de elementos <wsdl:Operation> válidos desde la descripción de servicio web que se utiliza para generar el archivo de enlace de servicio web. Cada

elemento <wsdl:Operation> está separado por un espacio; la lista puede abarcar más de una línea si es necesario. Puede utilizar este parámetro para los documentos WSDL 1.1 y WSDL 2.0.

PDSCP = *valor*

Especifica la página de códigos que se utiliza en los miembros del conjunto de datos particionados especificado en los parámetros **REQMEM** y **RESPMEM**, donde *valor* es un número CCSID o un número de página de códigos de Java. Si no se especifica este parámetro, se utiliza la página de códigos de los servicios del sistema z/OS UNIX. Por ejemplo, puede especificar **PDSCP** = 037.

PDSLIB = *valor*

Especifica el nombre del conjunto de datos particionados que contiene el lenguaje de alto nivel generado. Los miembros del conjunto de datos que utilizados para la solicitud y respuesta se especifican en los parámetros **REQMEM** y **RESPMEM** respectivamente.

PDSMEM = *valor*

Especifica el prefijo de 1 a 6 caracteres que utiliza DFHWS2LS para generar los nombres de los miembros del conjunto de datos particionados que contendrán las estructuras de lenguaje de alto nivel para tipos de datos abstractos. Genera el nombre de miembro añadiendo un número de dos dígitos al prefijo.

Utilice este parámetro en un nivel de correlación de 2.2 o superior para poner nombre a las estructuras de lenguaje asociadas a tipos de datos abstractos. Si se omite el parámetro **PDSMEM**, las estructuras de lenguaje para tipos de datos abstractos se nombran utilizando el valor del parámetro **REQMEM**.

PGMINT = { **CHANNEL** | **COMMAREA** }

Para un proveedor de servicios, especifique cómo CICS pasa datos al programa de aplicación de destino:

CHANNEL

CICS utiliza una interfaz de canal para pasar datos al programa de aplicación de destino.

COMMAREA

CICS utiliza un área de comunicación para pasar datos al programa de aplicación de destino.

Este parámetro se ignora cuando se utiliza la salida de DFHWS2LS en un solicitante de servicio.

Cuando el programa de aplicación de destino ha procesado la solicitud, debe utilizar el mismo mecanismo para devolver la respuesta. Si se ha recibido la solicitud en un área de comunicación, la respuesta debe devolverse en el área de comunicación; si la solicitud se ha recibido en un contenedor, la respuesta debe devolverse en un contenedor. La longitud del contenedor o área de comunicaciones que pasa CICS al programa de aplicación de destino es mayor que las longitudes del contenedor o área de comunicación de solicitud y el contenedor o área de comunicación de la respuesta.

PGMNAME = *valor*

Especifica el nombre de un recurso PROGRAM de CICS.

Cuando se utiliza DFHWS2LS para generar un archivo de enlace de servicio web que se utilizará en un proveedor de servicios, debe proporcionar este parámetro. Especifica el nombre de recurso del programa de aplicación que se expone como un servicio web.

Cuando se utiliza DFHWS2LS para generar un archivo de enlace de servicio web que se utilizará en un solicitante de servicio, omita este parámetro.

REQMEM = *valor*

Especifica el prefijo de 1 a 6 caracteres que utiliza DFHWS2LS para generar los nombres de los miembros del conjunto de datos particionados que contendrán las estructuras de lenguaje de alto nivel para la solicitud de servicio web.

- Para un proveedor de servicio, la solicitud de servicio web es la entrada al programa de aplicación.
- Para un solicitante de servicio, la respuesta de servicio web es la salida del programa de aplicación.

DFHWS2LS genera un miembro de conjunto de datos particionados para cada operación. Genera el nombre de miembro añadiendo un número de dos dígitos al prefijo.

Aunque este parámetro es opcional, debe especificarlo si la descripción de servicio web contiene una definición de una solicitud.

RESPMEM = *valor*

Especifica el prefijo de 1 a 6 caracteres que utiliza DFHWS2LS para generar los nombres de los miembros del conjunto de datos particionados que contendrán las estructuras de lenguaje de alto nivel para la respuesta de servicio web.

- Para un proveedor de servicios, la respuesta de servicio web es la salida del programa de aplicación.
- Para un solicitante de servicio, la respuesta de servicio web es la entrada al programa de aplicación.

DFHWS2LS genera un miembro de conjunto de datos particionados para cada operación. Genera el nombre de miembro añadiendo un número de dos dígitos al prefijo.

Omita este parámetro si no hay ninguna respuesta implicada; es decir, para mensajes unidireccionales.

SSL-KEYSTORE = *valor*

Este parámetro opcional especifica la ubicación completa del archivo de almacén de claves.

Utilice este parámetro si desea que el asistente de servicios web utilice el cifrado de capa de sockets seguros (SSL) para comunicarse a través de una red con IBM WebSphere Service Registry and Repository (WSRR).

SSL-KEYPWD = *valor*

Este parámetro opcional especifica la contraseña para el almacén de claves.

Utilice este parámetro si desea que el asistente de servicios web utilice el cifrado de capa de sockets seguros (SSL) para comunicarse a través de una red con IBM WebSphere Service Registry and Repository (WSRR).

SSL-TRUSTSTORE = *valor*

Este parámetro opcional especifica la ubicación completa del archivo de almacén de confianza.

Utilice este parámetro si desea que el asistente de servicios web utilice el cifrado de capa de sockets seguros (SSL) para comunicarse a través de una red con IBM WebSphere Service Registry and Repository (WSRR).

SSL-TRUSTPWD = *valor*

Este parámetro opcional especifica la contraseña para el almacén de confianza.

Utilice este parámetro si desea que el asistente de servicios web utilice el cifrado de capa de sockets seguros (SSL) para comunicarse a través de una red con IBM WebSphere Service Registry and Repository (WSRR).

STRUCTURE = (*solicitud* , *respuesta*)

Sólo para C y C++, especifica cómo se generan los nombres de las estructuras de solicitud y respuesta.

Las estructuras de solicitud y respuesta generadas reciben los nombres de *solicitud nn.* y *respuesta nn* donde *nn* es el sufijo numérico que se genera para distinguir las estructuras para cada operación.

Si se omiten uno o ambos nombres, las estructuras tienen el mismo nombre que los nombres de miembro de conjunto de datos particionados generados desde los parámetros **REQMEM** y **RESPMEM** que especifica.

SYNCONRETURN = { **NO** | **YES** }

Especifica si el servicio web remoto puede emitir un punto de sincronización.

NO El servicio web remoto no puede emitir un punto de sincronización. Este valor es el predeterminado. Si el servicio web remoto emite un punto de sincronización, falla con una terminación anómala ADPL.

YES El servicio web remoto puede emitir un punto de sincronización. Si selecciona **YES** , la tarea remota se confirma como una unidad de trabajo individual cuando se devuelve el control desde el servicio web remoto. Si el servicio web remoto actualiza un recurso recuperable y se produce un fallo después de que se devuelve, la actualización para ese recurso no se puede restituir.

TRANSACTION = *nombre*

En un proveedor de servicios, este parámetro especifica de uno a cuatro caracteres del nombre de una transacción de alias que puede iniciar la interconexión. El valor de este parámetro se utiliza para definir el atributo **TRANSACTION** del recurso **URIMAP** cuando se crea automáticamente utilizando el mandato de exploración **PIPELINE**.

Caracteres aceptables:

A-Z a-z 0-9 \$ @ # _ < >

URI = *valor*

En un proveedor de servicios, este parámetro especifica el URI relativo que utiliza un cliente para acceder al servicio web. CICS utiliza el valor que se especifica cuando genera un recurso **URIMAP** desde un archivo de enlace de servicio web que ha creado **DFHWS2LS**. El parámetro especifica el componente de vía de acceso del URI al que se aplica la definición **URIMAP**.

En una solicitud de servicio, el URI del servicio web de destino *no* se especifica con este parámetro. CICS no genera un recurso **URIMAP** para un solicitante de servicio. Puede definir su propio recurso **URIMAP** para que los solicitantes de servicio lo utilicen cuando hagan solicitudes de cliente al URI del servicio web de destino. Cuando un solicitante de servicio emite el mandato **INVOKE SERVICE**, CICS utiliza la ubicación **soap:address** desde el **wsdl:port** especificado en la descripción de servicio web si está presente. Puede sustituir eso y especificar un URI diferente utilizando las opciones **URIMAP** o **URI** en el mandato **INVOKE SERVICE**.

USERID = *id*

En un proveedor de servicios, este parámetro especifica un ID de usuario de 1 a 8 caracteres, que puede utilizar cualquier cliente web. Para una respuesta generada por la aplicación o un servicio web, la transacción de alias se conecta bajo este ID de usuario. El valor de este parámetro se utiliza para definir el atributo **USERID** del recurso **URIMAP** cuando se crea automáticamente utilizando el mandato de exploración **PIPELINE**.

Caracteres aceptables:

A-Z a-z 0-9 \$ @ #

WIDE-COMP3 = { FULL | NO | YES }

Controla el tamaño máximo de la longitud variable decimal empaquetada en la estructura de lenguaje COBOL o PL/I generada.

FULL Para COBOL y PL/I. DFHJS2LS genera un campo decimal empaquetado que es lo suficientemente grande como para contener todos los valores válidos. El tamaño máximo es de 31 dígitos. Es el valor predeterminado.

NO Solo para COBOL. DFHJS2LS limita la longitud variable de decimal empaquetado a 10 al generar el tipo de estructura de lenguaje COBOL COMP-3. Si el tamaño del decimal empaquetado es mayor de 18, se emite un mensaje DFHPI9022W para indicar que el tipo especificado se está restringiendo a un total de 18 dígitos.

YES Solo para COBOL. DFHJS2LS soporta el tamaño máximo de 31 al generar el tipo de estructura de lenguaje COBOL COMP-3.

Nota: Las opciones NO y YES generan campos que no pueden representar todos los valores válidos; la opción FULL evita este problema. Sin embargo, la opción FULL no permite que algunos valores no válidos se representen en el campo de decimal empaquetado. Por ejemplo, si un esquema indica que hay un máximo de cinco dígitos y un máximo de dos dígitos fraccionales, la opción FULL generará un campo de decimal empaquetado que permite siete dígitos, y esto permite espacio para valores válidos como 25000 y 999,99, pero también proporciona espacio para algunos valores no válidos como 9999,99. Cuando utilice la opción FULL, tenga cuidado de no generar valores no válidos en datos de aplicación.

WSADDR-EPR-ANY = { TRUE | FALSE }

Especifica si CICS transforma una referencia de punto final (EPR) de WS-Addressing en sus partes de componentes en las estructuras de lenguaje o trata la EPR como un tipo <xsd:any>. Tratar la EPR como un tipo <xsd:any> que la API de **WSACONTEXT BUILD** puede utilizar el EPR XML directamente.

FALSE

DFHWS2LS se comporta normalmente, transformando el XML en una estructura de lenguaje de alto nivel.

TRUE Establecer la opción en TRUE significa que en tiempo de ejecución CICS trata toda la EPR como un tipo <xsd:any> y coloca el XML de EPR en un contenedor al que puede hacer referencia la aplicación. La aplicación puede utilizar el XML de EPR con la API de **WSACONTEXT BUILD** para construir una EPR en el contexto de direccionamiento.

Este parámetro sólo está disponible en el nivel de tiempo de ejecución 3.0 y posteriores.

WSBIND = *valor*

El nombre completo del archivo de enlace de servicio web de z/OS UNIX. DFHWS2LS crea el archivo, pero no la estructura de directorio, si no existe. La extensión de archivo predeterminada es .wsbind.

WSDL = *valor*

El nombre completo del archivo de z/OS UNIX que contiene la descripción de servicio web. El nombre de archivo se restringe a los caracteres que son válidos

para una URL y, en particular, no pueden contener caracteres #. Si está utilizando WSRR para recuperar el documento WSDL, este parámetro especifica la ubicación en el sistema de archivos en el que se escribirá una copia local del documento WSDL.

WSDL-SERVICE = *valor*

Especifica el elemento `wsdl:Service` que se utiliza cuando la descripción de servicio web contiene más de un elemento Servicio para un elemento Enlace. Si especifica un valor para el parámetro **BINDING**, el elemento Servicio que especifica para este parámetro debe ser coherente con el elemento Enlace especificado. Puede utilizar este parámetro con los documentos WSDL 1.1 o WSDL 2.0.

WSRR-NAME = *valor*

Especifica el nombre del documento WSDL a recuperar de WSRR. Utilice este parámetro sólo cuando se especifique el parámetro **WSRR-SERVER**.

WSRR-NAMESPACE = *valor*

Especifica el espacio de nombres del documento WSDL que se va a recuperar de WSRR. También puede utilizar opcionalmente este parámetro cuando se especifica el parámetro **WSRR-SERVER** para calificar el nombre del documento WSDL especificado en el parámetro **WSRR-NAME**.

WSRR-PASSWORD = *valor*

Utilice este parámetro opcional si debe entrar una contraseña para acceder a WSRR.

Si se especifica el parámetro **WSRR-USERNAME**, también debe especificar este parámetro.

Utilice este parámetro sólo cuando se especifique el parámetro **WSRR-SERVER**.

WSRR-SERVER = { *nombre de dominio : número de puerto | dirección IP : número de puerto* }

Utilice este parámetro para especificar la ubicación del servidor IBM WebSphere Service Registry and Repository (WSRR). Si se especifica este parámetro, se utiliza la validación del parámetro WSRR.

WSRR-USERNAME = *valor*

Utilice este parámetro opcional si necesita especificar un nombre de usuario para acceder a WSRR. WSRR utiliza este nombre de usuario para establecer la propiedad de propietario.

Utilice este parámetro sólo cuando se especifique el parámetro **WSRR-SERVER**.

WSRR-VERSION = *valor*

Especifica la versión del documento WSDL a recuperar de WSRR. Puede utilizar este parámetro sólo cuando se especifica el parámetro **WSRR-SERVER**.

XML-ONLY = { **TRUE** | **FALSE** }

Especifica si CICS transforma o no el XML en el mensaje SOAP en datos de aplicación. Utilice el parámetro **XML-ONLY** para escribir aplicaciones de servicio web que procesan el XML ellos mismos.

TRUE CICS no realiza ninguna transformación en el XML. La aplicación de solicitante o proveedor de servicio debe trabajar con el contenido del contenedor DFHWS-BODY directamente para correlacionar datos entre el XML y el lenguaje de alto nivel.

FALSE

CICS transforma en XML en un lenguaje de alto nivel.

Este parámetro sólo está disponible en el nivel de tiempo de ejecución 2.1 y posteriores.

Otra información

- El ID de usuario bajo el que se ejecuta DFHWS2LS debe configurarse para utilizar los servicios de sistema UNIX. El ID de usuario debe tener permiso de lectura para la estructura de archivo de CICS z/OS UNIX y las bibliotecas PDS y permiso de escritura para los directorio especificados en los parámetros **LOGFILE**, **WSBIND** y **WSDL**.
- El ID de usuario debe tener una asignación de almacenamiento lo suficientemente grande para ejecutar Java.
- El JCL tiene una longitud de parámetro máxima de 100 caracteres. Esto puede aumentarse utilizando la sentencia **STDPARM**. Para obtener más información, consulte el apartado z/OS UNIX System Services User's Guide.

Ejemplo

```
//WS2LS JOB '  
información de contabilidad  
'  
,  
name,MSGCLASS=A  
// SET QT='''  
//JAVAPROG EXEC DFHWS2LS,  
// TMPFILE=&QT.&SYSUID.&QT  
//INPUT.SYSUT1 DD *  
PDSLIB=//CICSHLQ.SDFHSAMP  
REQMEM=CPYBK1  
RESPMEM=CPYBK2  
LANG=COBOL  
LOGFILE=/u/exampleapp/wsbinding/example.log  
MAPPING-LEVEL=3.0  
MAPPING-OVERRIDES=UNDERSCORES-AS-HYPHENS  
CHAR-VARYING=NULL  
INLINE-MAXOCCURS-LIMIT=2  
PGMNAME=DFH0XCMN  
URI=exampleApp/example  
PGMINT=COMMAREA  
SYNCONRETURN=YES  
WSBIND=/u/exampleapp/wsbinding/example.wsbinding  
WSDL=/u/exampleapp/wsd1/example.wsd1  
/*
```

Parámetros que necesita DFHWS2LS para soportar WS-Addressing:

Cuando configura el WSDL para Web Services Addressing, debe establecer los parámetros **MINIMUM-RUNTIME** y **MAPPING-LEVEL** en el asistente de servicios web, DFHWS2LS, en un valor de 3.0 o superior. Es posible que desee establecer el parámetro **WSADDR-EPR-ANY** en TRUE.

Establezca el parámetro **MINIMUM-RUNTIME** en el asistente de servicios web, DFHWS2LS, en 3.0 o superior. Un nivel de tiempo de ejecución de al menos 3.0 garantiza que cualquier archivo WSBinding que genera el asistente soporta completamente el direccionamiento de servicios web y puede interoperar con otras plataformas de servicios web.

Establezca el parámetro **MAPPING-LEVEL** en el asistente de servicios web, DFHWS2LS, en 3.0 o superior.

Si tiene cualquier elemento del tipo `wsa:EndpointReferenceType` en los mensajes de solicitud o respuesta definidos en el documento WSDL, y desea utilizar esos

elementos como entrada para el mandato de API **WSACONTEXT BUILD** en tiempo de ejecución, establezca el parámetro **WSADDR-EPR-ANY** en TRUE. Establecer el parámetro **WSADDR-EPR-ANY** en TRUE indica que CICS no debe transformar la EPR en una estructura de lenguaje en tiempo de ejecución; en lugar de eso, CICS debe tratar los datos de EPR como un elemento `<xsd:any>` y almacenarlo en un contenedor con nombre.

Este fragmento WSDL de ejemplo muestra un MAP `<wsa:To>` que se está pasando como elemento de tipo `wsa:EndpointReferenceType`:

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="exampleEPR" targetNamespace="http://example.ibm.com/"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:s0="http://example.ibm.com/"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata">
  <types>
    <xs:schema targetNamespace="http://test.org/"
      xmlns:s="http://www.w3.org/2001/XMLSchema"
      xmlns:s0="http://example.ibm.com/"
      xmlns:wsa="http://www.w3.org/2005/08/addressing">
      ...
      <xs:element name="exampleResponse" type="s0:typeResponse"/>
      <xs:complexType name="typeResponse">
        <xs:sequence>
          <xs:element name="myEpr" type="wsa:EndpointReferenceType"/> 1
        </xs:sequence>
      </xs:complexType>
      ...
    </xs:schema>
  </types>
  ...
  <message name="msgResponse">
    <part element="s0:exampleResponse" name="response"/>
  </message>
  ...
</definitions>
```

Cuando DFHWS2LS procesa el elemento `<xs:element name="myEpr" type="wsa:EndpointReferenceType"/>` **1**, con el parámetro **WSADDR-EPR-ANY** establecido en TRUE, los datos de elemento `myEpr` se almacenarán en el contenedor con nombre como un elemento `<xsd:any>` en tiempo de ejecución y un puntero al contenedor añadido a la estructura de lenguaje generada.

Por ejemplo, la estructura de lenguaje COBOL generada por DFHWS2LS para el elemento `myEpr` se muestra aquí:

```
09 myEpr.
   12 myEpr-xml-cont          PIC X(16).
   12 myEpr-xmlns-cont       PIC X(16).
```

El contenedor `myEpr-xml-cont` almacena el nombre del contenedor que posee los datos `myEpr`. `myEpr-xmlns-cont` es un contenedor opcional que se llena con declaraciones de espacio de nombres XML que están en el ámbito.

Niveles de correlación para los asistentes de CICS

Una correlación es un conjunto de reglas que especifica cómo se convierte la información entre estructuras de lenguajes y esquemas XML. Para beneficiarse de las correlaciones más sofisticadas disponibles, debe establecer el parámetro **MAPPING-LEVEL** en los asistentes de CICS en el nivel más reciente.

Cada nivel de correlación hereda la función de la correlación anterior, donde el nivel de correlación más alto ofrece las mejores prestaciones disponibles. El nivel de correlación más alto proporciona más control sobre la conversión de datos en tiempo de ejecución y elimina restricciones en el soporte para determinados tipos de datos y elementos XML.

Puede establecer el parámetro **MAPPING-LEVEL** en un nivel anterior si desea volver a desplegar las aplicaciones que se habían habilitado previamente en ese nivel.

Nivel de correlación 4.1

El nivel de correlación 4.1 es compatible con una región de CICS TS 5.3 o una región de CICS TS 5.2 con APAR PI67641 aplicado y superior.

El nivel de correlación 4.1 se añade a los asistentes de servicios web, los asistentes XML y asistentes de JSON. Este nivel de correlación implementa correlaciones mejoradas para matrices sencillas generadas de abajo arriba a partir de libros de copias existentes; también añade la capacidad de CICS para detectar automáticamente esos registros a partir del formato XML/JSON generado. Para obtener más información, consulte el apartado “Correlación de COBOL con esquemas JSON” en la página 436.

DFHLS2WS, DFHWS2LS, DFHLS2SC, DFHSC2LS, DFHJS2LS y DFHLS2JS soportan los parámetros **TRUNCATE-NULL-ARRAYS** y **TRUNCATE-NULL-ARRAY-VALUES**.

Si especifica cualquier valor para **TRUNCATE-NULL-ARRAY-VALUES**, también debe especificar **TRUNCATE-NULL-ARRAYS=ENABLED**.

Nivel de correlación 4.0

El nivel de correlación 4.0 es compatible con una región CICS TS 5.2.

En el nivel de correlación 4.0 y superior, DFHLS2SC y DFHLS2WS soportan la cláusula COBOL OCCURS DEPENDING ON y la correlación de matrices de carácter COBOL en series XML. Puede establecer este comportamiento utilizando el parámetro **CHAR-OCCURS** en los asistentes de CICS.

- Debe especificar el parámetro **DATA-TRUNCATION=ENABLED**.
- Las cláusulas OCCURS DEPENDING ON complejas no son compatibles. Esta limitación significa que OCCURS DEPENDING ON solo es válida para el último campo de una estructura.
- CICS no soporta nombres completos (que utilizan la palabra clave 'OF') como destino de una cláusula OCCURS DEPENDING ON, por ejemplo FIELD1 OF STRUCTURE1.
- CICS no soporta la palabra clave UNBOUNDED. Debe especificar el tamaño máximo de la tabla que está esperando la aplicación.

En el nivel de correlación 4.0 y superior, los servicios web de CICS soportan la conversión de datos de aplicación que se codifican utilizando UTF-16 Unicode.

- Cuando utiliza LS2WS o LS2SC, puede habilitar este comportamiento utilizando tipos de datos específicos del lenguaje para UTF-16.
- Cuando utiliza WS2LS o SC2LS, puede habilitar este comportamiento estableciendo CCSID=1200.
- CICS soporta únicamente una única página de códigos Unicode, “ UTF-16BE con IBM Private Use Area ” (CCSID 1200).

- No se soporta la conversión de datos de aplicación que se codifican utilizando UTF-8.

Nota: DFHLS2WS y DFHLS2SC no soportan la cláusula COBOL GROUP USAGE NATIONAL.

Nivel de correlación 3.0 y posterior

El nivel de correlación 3.0 es compatible con una región de CICS TS 4.1 y superior.

Este nivel de correlación ofrece las siguientes compatibilidades:

- DFHSC2LS y DFHWS2LS correlacionan tipos de datos `xsd:dateTime` con formato CICS ASKTIME.
- DFHLS2WS puede generar un documento WSDL y un enlace de servicio web desde una aplicación que utiliza muchos contenedores en lugar de sólo un contenedor.
- Soporte de datos truncados que se describen por medio de una estructura de datos de longitud fija. Puede establecer este comportamiento utilizando el parámetro **DATA-TRUNCATION** en los asistentes de CICS.

Nivel de correlación 2.2 y posterior

El nivel de correlación 2.2 es compatible con una región de CICS TS 3.2, con APAR PK69738 aplicado y superior.

En el nivel de correlación 2.2 y superior, DFHSC2LS y DFHWS2LS soportan las siguientes correlaciones XML:

- Valores fijos para elementos
- Grupos de sustitución
- Tipos de datos abstractos
- Los elementos `<sequence>` del esquema XML puede anidar dentro de elementos `<choice>`

DFHSC2LS y DFHWS2LS proporcionan un soporte mejorado para las siguientes correlaciones XML:

- Elementos abstractos
- Elementos `<choice>` del esquema XML

Nivel de correlación 2.1 y posterior

El nivel de correlación 2.1 es compatible con una región de CICS TS 3.2, con APAR PK59794 aplicado y superior.

Este nivel de correlación incluye un mayor control sobre la forma en la que se maneja el contenido de la variable con el nuevo parámetro **INLINE-MAXOCCURS-LIMIT** y los nuevos valores en el parámetro **CHAR-VARYING**.

En el nivel de correlación 2.1 y superior, DFHSC2LS y DFHWS2LS ofrecen el siguiente soporte nuevo y mejorado para correlaciones XML:

- El elemento `<any>` del esquema XML
- El tipo `xsd:anyType`
- Tolerancia de elementos abstractos
- El parámetro **INLINE-MAXOCCURS-LIMIT**

- El atributo minOccurs

El parámetro **INLINE-MAXOCCURS-LIMIT** especifica si las listas de repetición variable se correlacionan en línea. Para obtener más información sobre la correlación del contenido que se repite de forma variable en línea, consulte Matrices variables de los elementos.

El soporte para el atributo minOccurs se mejora para los elementos <sequence>, <choice> y <all> del esquema XML. Si minOccurs="0" , el asistente de CICS trata estos elementos como si el atributo minOccurs="0" fuera también atributo de todos sus elementos hijo.

En el nivel de correlación 2.1 y superior, DFHLS2SC y DFHLS2WS soportan las siguientes correlaciones XML:

- No se tienen en cuenta los campos FILLER en COBOL y PL/I
- Un valor de COLLAPSE para el parámetro **CHAR-VARYING**
- Un valor de BINARY para el parámetro **CHAR-VARYING**

Los campos FILLER en COBOL y PL/I se ignoran; no aparece en el esquema XML generado y queda un hueco adecuado en las estructuras de datos en el tiempo de ejecución.

COLLAPSE hace que CICS ignore los espacios finales en los campos de texto.

BINARY proporciona soporte para los campos binarios. Este valor es útil cuando está convirtiendo COBOL en un esquema XML. Esta opción está disponible sólo en matrices de caracteres SBCS y permite que la matriz se correlacione con campos xsd:base64Binary de longitud fija en lugar de a campos xsd:string.

Nivel de correlación 1.2 y superior

El nivel de correlación 1.2 es compatible con una región de CICS TS 3.1 y superior.

Hay disponible un mayor control sobre la manera en la que se transforman los datos binarios y los caracteres en el tiempo de ejecución con estos parámetros adicionales en las herramientas por lotes:

- **CHAR-VARYING**
- **CHAR-VARYING-LIMIT**
- **CHAR-MULTIPLIER**
- **DEFAULT-CHAR-MAXLENGTH**

Si decide utilizar el parámetro **CHAR-MULTIPLIER** en DFHSC2LS o DFHWS2LS, las reglas siguientes se aplican después de que se utiliza el valor de este parámetro para calcular la cantidad de espacio necesario para los datos de caracteres.

- DFHSC2LS y DFHWS2LS proporcionan estas correlaciones:
 - Tipos de datos de caracteres de longitud variable que tienen una longitud máxima de más de 32 767 bytes correlacionados con un contenedor. Puede utilizar el parámetro **CHAR-VARYING-LIMIT** para establecer un límite inferior. Se crea un campo de 16 bytes en la estructura de lenguaje para almacenar el nombre del contenedor. En el tiempo de ejecución, los datos de caracteres se almacenan en un contenedor y el nombre del contenedor se coloca en la estructura de lenguaje.

- Los tipos de datos de caracteres de longitud variable que tienen una longitud máxima de menos de 32 768 bytes se correlacionan con una estructura VARYING para todos los lenguajes excepto C/C++ y Enterprise PL/I. En C/C++, estos tipos de datos se correlacionan con series terminadas en nulo, y en Enterprise PL/I estos tipos de datos se correlacionan con estructuras VARYINGZ. Puede utilizar el parámetro **CHAR-VARYING** para seleccionar la forma en la que se correlacionan los datos de caracteres de longitud variable.
- Los datos binarios de longitud variable que tienen una longitud máxima de menos de 32 768 bytes se correlacionan con una estructura VARYING para todos los lenguajes. Si la longitud máxima es igual o mayor de 32 768 bytes, los datos se correlacionan con un contenedor. Se crea un campo de 16 bytes en la estructura de lenguaje para almacenar el nombre del contenedor. En el tiempo de ejecución, los datos binarios se almacenan en un contenedor y el nombre del contenedor se coloca en la estructura de lenguaje.

Si tiene tipos de datos de caracteres en el esquema XML que no tienen una longitud asociada a ellos, puede asignar una longitud predeterminada utilizando el parámetro **DEFAULT-CHAR-MAXLENGTH** en DFHWS2LS o DFHSC2LS.

DFHLS2SC y DFHLS2WS proporcionan estas correlaciones:

- Los campos de caracteres se correlacionan con un tipo de datos xsd:string y se pueden procesar como campos de longitud fija o series terminadas en nulo en tiempo de ejecución. Puede utilizar el parámetro **CHAR-VARYING** para seleccionar la forma en la que los datos de caracteres de longitud variable se manejan en tiempo de ejecución para todos los lenguajes excepto PL/I.
- Los tipos de datos Base64Binary se correlacionan con un contenedor si la longitud máxima de los datos es mayor de 32 767 bytes o cuando la longitud no está definida. Si la longitud de los datos es 32 767 o menos, el tipo de datos base64Binary se correlaciona con una estructura VARYING para todos los lenguajes.

Nivel de correlación 1.1 y posterior

El nivel de correlación 1.1 es compatible con una región de CICS TS 3.1 y superior.

Este nivel de correlación proporciona una correlación mejorada de tipos de datos binarios y caracteres XML, en particular cuando se correlacionan datos de longitud variable que tienen los atributos `maxLength` y `minLength` definidos con diferentes valores en el esquema XML. Los datos se manejan de la siguiente manera:

- Los tipos de datos binarios y de caracteres que tienen una longitud fija que son mayores de 16 MB se correlacionan con un contenedor para todos los lenguajes excepto PL/I. En PL/I, los tipos de datos binarios y de caracteres de longitud fija que son mayores de 32 767 se correlacionan con un contenedor. Se crea un campo de 16 bytes en la estructura de lenguaje para almacenar el nombre del contenedor. En el tiempo de ejecución, los datos de longitud fija se almacenan en un contenedor y el nombre del contenedor se coloca en la estructura de lenguaje. Puesto que los contenedores son variables en longitud, los datos de longitud fija que se correlacionan con un contenedor no se rellenan con espacios o nulos, o se truncan, para coincidir con la longitud fija especificada en el esquema XML o la descripción de servicio web. Si la longitud de los datos es significativa, puede escribir su aplicación para comprobarla o activar la validación en la región de CICS. Las validaciones SOAP y XML tienen un impacto de rendimiento significativo.

- Los tipos de datos <list> y <union> del esquema XML se correlacionan con campos de caracteres.
- Los atributos XML definidos por el esquema se correlacionan en lugar de ignorarse. Se permite un máximo de 255 atributos para cada elemento XML. Para obtener más información, consulte Soporte para atributos XML.
- Se soporta el atributo `xsi:nil`. Para obtener más información, consulte Soporte para atributos XML.

Solo nivel de correlación 1.1

El nivel de correlación 1.1 es compatible con una región de CICS TS 3.1 y superior.

Este nivel de correlación proporciona una correlación mejorada de tipos de datos binarios y caracteres XML, en particular cuando se correlacionan datos de longitud variable que tienen los atributos `maxLength` y `minLength` definidos con diferentes valores en el esquema XML. Los datos se manejan de la siguiente manera:

- Los tipos de datos binarios de longitud variable se correlacionan con un contenedor. Se crea un campo de 16 bytes en la estructura de lenguaje para almacenar el nombre del contenedor. En el tiempo de ejecución, los datos binarios se almacenan en un contenedor y el nombre del contenedor se coloca en la estructura de lenguaje.
- Tipos de datos de caracteres de longitud variable que tienen una longitud máxima de más de 32 767 bytes se correlacionan con un contenedor. Se crea un campo de 16 bytes en la estructura de lenguaje para almacenar el nombre del contenedor. En el tiempo de ejecución, los datos de caracteres se almacenan en un contenedor y el nombre del contenedor se coloca en la estructura de lenguaje.
- Los tipos de datos binarios y de caracteres que tienen una longitud fija de menos de 16 MB se correlacionan con campos de longitud fija para todos los lenguajes excepto PL/I. En PL/I, los tipos de datos binarios y de caracteres de longitud fija de 32 767 o menos se correlacionan con campos de longitud fija.
- CICS codifica y decodifica datos en formato `hexBinary` pero no en formato `base64Binary`. Los tipos de datos `Base64Binary` en el esquema XML se correlacionan con un campo en la estructura de lenguaje. El tamaño del campo se calcula utilizando la fórmula: $4 \times (\text{ceil}(z / 3))$ donde:
 - z es la longitud del tipo de datos en el esquema XML.
 - $\text{ceil}(x)$ es el entero más pequeño mayor que o igual a x .

Si la longitud de z es mayor de 24 566 bytes, la estructura de lenguaje resultante no puede compilarse. Si tiene datos `base64Binary` que son más grandes de 24 566 bytes, debe utilizar un nivel de correlación de 1.2. Con el nivel de correlación 1.2, puede correlacionar los datos `base64Binary` con un contenedor en lugar de utilizar un campo en la estructura de lenguaje.

Solo nivel de correlación 1.0

El nivel de correlación 1.0 es compatible con una región de CICS TS 3.1 y superior.

Tenga en cuenta las siguientes limitaciones, que se modifican en niveles de correlación posteriores:

- DFHSC2LS y DFHWS2LS correlacionan tipos de datos binarios y de caracteres en el esquema XML con campos de longitud fija en la estructura del lenguaje. Mire este esquema XML parcial:


```

<xsd:element name="example">
<xsd:simpleType>
<xsd:restriction base="xsd:string">
<xsd:maxLength value="33000"/>
</xsd:restriction>
</xsd:simpleType>
</xsd:element>

```

Este esquema XML parcial aparece en una estructura de lenguaje COBOL como este ejemplo:

```
15 example PIC X(33000)
```

- CICS codifica y decodifica datos en formato hexBinary pero no en formato base64Binary. DFHSC2LS y DFHWS2LS correlacionan datos Base64Binary con un campo de caracteres de longitud fija, el contenido de los cuales debe codificarse o descodificarse mediante el programa de aplicación.
- DFHSC2LS y DFHWS2LS ignoran los atributos XML durante el proceso.
- DFHLS2SC y DFHLS2WS interpretan los campos binarios y de caracteres en la estructura de lenguaje como campos de longitud fija y correlacionan esos campos con elementos XML que tienen un atributo maxLength. En tiempo de ejecución, los campos de la estructura de lenguaje se rellenan con espacios o nulos si no hay datos suficientes.

Correlación de lenguaje de alto nivel y esquema XML:

Uso de los asistentes de CICS para generar correlaciones entre estructuras de lenguaje de alto nivel y esquemas XML o documentos WSDL. Los asistentes de CICS también generan esquemas XML o documentos WSDL de estructuras de datos de lenguaje de alto nivel o viceversa.

Los programas de utilidad DFHSC2LS y DFHLS2SC se conocen colectivamente como el asistente de CICS XML. Los programas de utilidad DFHWS2LS y DFHLS2WS se conocen colectivamente como el asistente de servicios web de CICS.

- DFHLS2SC y DFHLS2WS correlacionan estructuras de lenguaje de alto nivel a esquemas XML y documentos WSDL respectivamente.
- DFHSC2LS y DFHWS2LS correlacionan esquemas XML y documentos WSDL a estructuras de lenguaje de alto nivel.

Las dos correlaciones no son simétricas:

- Si procesa una estructura de datos de lenguaje con DFHLS2SC o DFHLS2WS y, a continuación, procesa el esquema XML o documento WSDL resultante con el programa de utilidad complementario (DFHSC2LS o DFHWS2LS respectivamente), no espere que la estructura de datos final sea la misma que el original. Sin embargo, la estructura de datos final es equivalente lógicamente al original.
- Si procesa un esquema XML o documento WSDL con DFHSC2LS o DFHWS2LS y, a continuación, procesa la estructura de lenguaje resultante con el programa de utilidad complementario (DFHLS2SC o DFHLS2WS respectivamente), no espere que el esquema XML en el esquema XML o documento WSDL final sea el mismo que el original.
- En algunos casos, DFHSC2LS y DFHWS2LS generan estructuras de lenguaje que no soportan DFHLS2SC y DFHLS2WS.

Debe codificar las estructuras de lenguaje procesadas por DFHLS2SC y DFHLS2WS según las reglas del lenguaje, como se implementan en los compiladores de lenguaje que soporta CICS.

Correlación de COBOL con esquemas JSON:

El programa de utilidad DFHLS2JS soporta correlaciones entre estructuras de datos COBOL y definiciones de esquema JSON.

Los nombres COBOL se convierten en nombres JSON según las reglas siguientes:

- Los nombres duplicados se convierten en exclusivos añadiendo uno o varios dígitos numéricos.

Por ejemplo, dos instancias de year pasan a ser year y year1.

- Los guiones se sustituyen por guiones bajos. Las series de guiones contiguos son cambiados por guiones bajos contiguos.

Por ejemplo, current-user--id se convierte en current_user__id.

- Los segmentos de nombres que están delimitados por guiones y que sólo contienen caracteres en mayúsculas se convierten en minúsculas.

Por ejemplo, CA-REQUEST-ID pasa a ser ca_request_id.

- Se añade un guión bajo inicial a los nombres que empiezan por un carácter numérico.

Por ejemplo, 9A-REQUEST-ID pasa a ser _9a_request_id.

CICS correlaciona elementos de descripción de datos COBOL con elementos de esquema de acuerdo con la tabla siguiente. Los elementos de descripción de datos de COBOL que no se muestran en la tabla no son compatibles con DFHLS2JS.

También se aplican las siguientes restricciones:

- Los elementos de descripción de datos con números de nivel de 66 y 77 no se soportan. Los elementos de descripción de datos con un número de nivel de 88 se ignoran.
- No se soportan las siguientes cláusulas en entradas de descripción de datos:
 - REDEFINES
 - RENAMES; es el nivel 66
 - DATE FORMAT
- Se ignoran las siguientes cláusulas en los elementos de descripción de datos:
 - BLANK WHEN ZERO
 - JUSTIFIED
 - VALUE
- Se soporta la cláusula SIGN "SIGN TRAILING". SIGN LEADING de la cláusula SIGN solo se admite cuando el nivel de correlación especificado en DFHLS2JS es 1.2 o superior.
- Se soporta SEPARATE CHARACTER en un nivel de correlación de 1.2 o superior tanto para la cláusula SIGN TRAILING como para SIGN LEADING.
- No se soportan las siguientes frases en la cláusula USAGE:
 - OBJECT REFERENCE
 - POINTER
 - FUNCTION-POINTER
 - PROCEDURE-POINTER
- Se soportan las siguientes frases en la cláusula USAGE en un nivel de correlación de 1.2 o superior:
 - COMPUTATIONAL-1
 - COMPUTATIONAL-2

- Los únicos caracteres PICTURE que se admiten para los elementos de descripción de datos DISPLAY y COMPUTATIONAL-5 son 9, S y Z.
- Los caracteres PICTURE que se admiten para los elementos de descripción de datos PACKED-DECIMAL son 9, S, V y Z.
- Los únicos caracteres PICTURE que se admiten con los elementos de descripción de datos numéricos editados son 9 y Z.
- Si el parámetro MAPPING-LEVEL se establece en 1.2 o superior y el parámetro CHAR-VARYING se establece en NULL, las matrices de caracteres se correlacionan con una string y se procesan como series terminadas en nulo.
- Si el parámetro MAPPING-LEVEL se establece en 1.2 o superior y el parámetro CHAR-VARYING se establece en BINARY, las matrices de caracteres se correlacionan con una string y se procesan como datos binarios.
- Si el parámetro MAPPING-LEVEL se establece en 1.2 o superior y el parámetro CHAR-VARYING se establece en COLLAPSE, se ignora el espacio en blanco final en las series.
- La cláusula OCCURS DEPENDING ON es compatible a nivel de correlación 4.0 o superior. Las cláusulas OCCURS DEPENDING ON complejas no son compatibles. Esto significa que OCCURS DEPENDING ON sólo se soporta para el último campo de una estructura.
- La cláusula OCCURS INDEXED BY es compatible a nivel de correlación.

Descripción de datos COBOL	Definición de esquema JSON
PIC X(<i>n</i>) PIC A(<i>n</i>) PIC G(<i>n</i>) DISPLAY-1 PIC N(<i>n</i>)	<pre>"type": "string", "maxLength": <i>n</i></pre>

Descripción de datos COBOL	Definición de esquema JSON
PIC S9 DISPLAY PIC S99 DISPLAY PIC S999 DISPLAY PIC S9999 DISPLAY PIC S9(<i>n</i>) DISPLAY PIC S9(<i>n</i>) COMP PIC S9(<i>n</i>) COMP-4 PIC S9(<i>n</i>) COMP-5 PIC S9(<i>n</i>) BINARY	<pre>"type":"integer", "minimum":- (<i>n</i> + 1), "maximum": <i>n</i></pre> <p>donde <i>n</i> es el valor máximo que se puede representar mediante el patrón de '9' caracteres.</p>
PIC 9 DISPLAY PIC 99 DISPLAY PIC 999 DISPLAY PIC 9999 DISPLAY PIC 9(<i>n</i>) DISPLAY PIC 9(<i>n</i>) COMP PIC 9(<i>n</i>) COMP-4 PIC 9(<i>n</i>) COMP-5 PIC 9(<i>n</i>) BINARY	<pre>"type":"integer", "minimum":0, "maximum": <i>n</i></pre> <p>donde <i>n</i> es el valor máximo que puede ser representado por el patrón de '9' caracteres.</p>

Descripción de datos COBOL	Definición de esquema JSON
PIC S9(<i>m</i>)V9(<i>n</i>) COMP-3	<pre>"type": "number", "format": "decimal", "minimum": x , "maximum": y , "multipleOf": z</pre> <p>donde:</p> <p><i>x</i> es el valor mínimo que se puede representar mediante el patrón de '9' caracteres</p> <p><i>y</i> es el valor máximo que se puede representar mediante el patrón de '9' caracteres</p> <p><i>z</i> es la unidad más pequeña disponible = 1 / 10 "</p>
PIC 9(<i>m</i>)V9(<i>n</i>) COMP-3	<pre>"type": "number", "format": "decimal", "minimum": 0, "maximum": y , "multipleOf": z</pre> <p>donde:</p> <p><i>y</i> es el valor máximo que se puede representar mediante el patrón de '9' caracteres</p> <p><i>z</i> es la unidad más pequeña disponible = 1 / 10 "</p>
PIC S9(15) COMP-3 Admitido en el nivel de correlación 3.0 cuando DATETIME=PACKED15	<pre>"type": "string", "format": "date-time"</pre> <p>El formato de la indicación de fecha y hora se define por RFC3339.</p>
PIC S9(<i>m</i>)V9(<i>n</i>) DISPLAY Se soporta en el nivel de correlación 1.2 y superior	<pre>"type": "number", "format": "decimal", "minimum": x , "maximum": y , "multipleOf": z</pre> <p>donde:</p> <p><i>x</i> es el valor mínimo que se puede representar mediante el patrón de '9' caracteres</p> <p><i>y</i> es el valor máximo que se puede representar mediante el patrón de '9' caracteres</p> <p><i>z</i> es la unidad más pequeña disponible = 1 / 10 "</p>

Descripción de datos COBOL	Definición de esquema JSON
<p>COMP-1</p> <p>Se soporta en el nivel de correlación 1.2 y superior</p> <p>Nota: La representación de datos de coma flotante hexadecimal (HFP) de IBM no es exactamente lo mismo que la representación IEEE-754-1985 utilizada para JSON. Algunos valores podrían no convertirse de forma exacta de una representación a la otra.</p> <p>Algunos valores sumamente grandes o pequeños podrían no ser válidos para los tipos de datos flotantes. Algunos valores podrían perder precisión cuando se convierten a o desde la representación HFP. Si las conversiones precisas son importantes, considere sustituir el uso de tipos de datos COMP-1 por alternativas de precisión fija.</p>	<pre>"type": "number", "format": "float"</pre>
<p>COMP-2</p> <p>Se soporta en el nivel de correlación 1.2 y superior</p> <p>Nota: La representación de datos de coma flotante hexadecimal (HFP) de IBM no es exactamente lo mismo que la representación IEEE-754-1985 utilizada para JSON. Algunos valores podrían no convertirse de forma exacta de una representación a la otra.</p> <p>Algunos valores sumamente grandes o pequeños podrían no ser válidos para los tipos de datos dobles. Algunos valores podrían perder precisión cuando se convierten a o desde la representación HFP. Si las conversiones precisas son importantes, considere sustituir el uso de tipos de datos COMP-2 por alternativas de precisión fija. .</p>	<pre>"type": "number", "format": "double"</pre>

Descripción de datos COBOL	Definición de esquema JSON
<i>descripción de datos</i> OCCURS <i>n</i> TIMES	<p>En el nivel de correlación 4.0 e inferior</p> <p>Para primitivos:</p> <pre> "type": "array" "maxItems": n "minItems": n "items": { "type": "object", "properties": { name : { data description JSON } } "required": [name] } </pre> <p>Para estructuras:</p> <pre> "type": "array" "maxItems": n "minItems": n "items": { data description JSON } </pre> <p>Donde <i>JSON de descripción de datos</i> es la representación del esquema JSON de la <i>descripción de datos</i> COBOL y <i>name</i> es el nombre de la <i>descripción de datos</i> COBOL.</p> <p>En el nivel de correlación 4.1 y superior:</p> <p>TRUNCATE-NULL-ARRAYS = DISABLED</p> <p>Ambas correlaciones estructurada y primitiva se correlacionan de la siguiente manera.</p> <pre> "type": "array" "maxItems": n "minItems": n "items": { data description JSON } </pre> <p>TRUNCATE-NULL-ARRAYS = ENABLED</p> <p>Las correlaciones primitivas se correlacionan como se indica arriba y las correlaciones estructuras se correlacionan de la siguiente manera.</p> <pre> "type": "array" "maxItems": n "minItems": 0 "items": { data description JSON } </pre>

Descripción de datos COBOL	Definición de esquema JSON
<i>descripción de</i> <i>datos</i> OCCURS <i>n</i> TO <i>m</i> TIMES DEPENDING ON <i>t</i> Se admite en el nivel de correlación 4.0	<pre>"field-name" : { "type": "array" , "maxItems": <i>m</i> "minItems": <i>n</i> "items": { . . . } }</pre> El contenido del elemento de matriz depende del tipo de datos utilizado.
PIC X OCCURS <i>n</i> TIMES PIC A OCCURS <i>n</i> TIMES PIC G DISPLAY-1 OCCURS <i>n</i> TIMES PIC N OCCURS <i>n</i> TIMES	<p>Cuando CHAR-OCCURS =STRING :</p> <pre>"field-name" : { "type": "string" , "maxLength": <i>n</i> }</pre> Esta es una serie.

Descripción de datos COBOL	Definición de esquema JSON
	<p>Cuando CHAR-OCCURS =ARRAY :</p> <p>En el nivel de correlación 4.0 e inferior</p> <pre> "field-name" :{ "maxItems": m , "minItems": n , "items":{ "type": "object" , "properties":{ "field-name":{ "type": "string" , "maxLength":1 } }, "required":["field-name"] } } </pre> <p>Es una matriz de caracteres individuales.</p> <p>En el nivel de correlación 4.1 y superior:</p> <p>TRUNCATE-NULL-ARRAYS = DISABLED</p> <p>Ambas correlaciones estructurada y primitiva se correlacionan de la siguiente manera.</p> <pre> "type":"array" "maxItems": n "minItems": n "items":{ data description JSON } </pre> <p>TRUNCATE-NULL-ARRAYS = ENABLED</p> <p>Las correlaciones primitivas se correlacionan como se indica arriba y las correlaciones estructuras se correlacionan de la siguiente manera.</p> <pre> "type":"array" "maxItems": n "minItems": 0 "items":{ data description JSON } </pre>

Descripción de datos COBOL	Definición de esquema JSON
PIC X OCCURS <i>n</i> TO <i>m</i> TIMES DEPENDING ON <i>t</i> PIC A OCCURS <i>n</i> TO <i>m</i> TIMES DEPENDING ON <i>t</i> PIC G DISPLAY-1 OCCURS <i>n</i> TO <i>m</i> TIMES DEPENDING ON <i>t</i> PIC N OCCURS <i>n</i> TO <i>m</i> TIMES DEPENDING ON <i>t</i>	Cuando CHAR-OCCURS =STRING : "field-name" : { "type": "string" , "maxLength": <i>m</i> "minLength": <i>n</i> }
PIC N(<i>n</i>) USAGE NATIONAL Cuando CHAR-USAGE =NATIONAL : PIC N(<i>n</i>)	En el nivel de correlación 4.0 y superior: "type": "string", "maxLength": <i>n</i> En tiempo de ejecución, CICS rellena el campo de estructura de datos de aplicación con datos UTF-16.

Correlación de esquema JSON con COBOL:

El programa de utilidad DFHJS2LS soporta correlaciones entre esquemas JSON y estructuras de datos COBOL.

Los asistentes de CICS generan nombres de campo exclusivos y válidos para variables COBOL de los nombres de elemento de esquema que utilizan las reglas siguientes:

1. Las palabras reservadas de COBOL llevan el prefijo ' X ' inicial.
Por ejemplo, DISPLAY pasa a ser XDISPLAY.
2. Caracteres que no sean A-Z, a-z, 0-9 o guión se sustituyen por ' X ' .
Por ejemplo, monthly_total pasa a ser monthlyXtotal.
3. Si el último carácter es un guión se sustituye por ' X ' .
Por ejemplo, ca-request- pasa a ser ca-requestX.

- Los nombres duplicados en el mismo ámbito se convierten en exclusivos añadiendo uno o dos dígitos numéricos a la segunda y siguientes instancias del nombre.

Por ejemplo, tres instancias de year se convierten en year , year1 y year2.

- Un esquema JSON especifica que una variable tiene una cardinalidad variable si tiene un valor "type" de "array" , y las palabras claves "minItems" y "maxItems" se omiten o tienen diferentes valores. Si el esquema especifica que la variable tiene una cardinalidad variable, los nombres de campo se crean con los sufijos "_cont" y "_num".

Para obtener más información, consulte el apartado “Matrices de variables de elementos en DFHJS2LS” en la página 283.

- Un esquema JSON especifica que una variable es opcional si no aparece en la matriz de palabra clave "required" que está asociada al tipo "object" del esquema JSON delimitado. Para los campos opcionales, se genera un campo adicional con un sufijo _num añadido al nombre de elemento. En el tiempo de ejecución, es cero para indicar que falta el valor de los datos JSON, y no cero si el valor está presente en los datos JSON.
- Los nombres de campo están limitados a 28 caracteres. Si un nombre generado, incluido cualquier prefijo o sufijo, sobrepasa esta longitud, el nombre de elemento se trunca.

DFHJS2LS correlaciona tipos de esquema con elementos de descripción de datos COBOL utilizando el nivel de correlación especificado según la tabla siguiente. Tenga en cuenta los puntos siguientes:

- Si el parámetro **MAPPING-LEVEL** se establece en 1.2 o superior y el parámetro **CHAR-VARYING** se establece en NULL, los datos de caracteres de longitud variable se correlacionan con series terminadas en nulo y se asigna un carácter adicional al terminador en nulo.
- Si el parámetro **MAPPING-LEVEL** se establece en 1.2 o superior y el parámetro **CHAR-VARYING** se establece en YES, los datos de carácter de longitud variable se correlacionan con dos elementos relacionados: un campo de longitud y un campo de datos. Por ejemplo:

```
"textString": {  
  "type": "string",  
  "maxLength": 10000,  
  "minLength": 1  
}
```

se correlaciona con:

```
15 textString-length PIC S9999 COMP-5 SYNC  
15 textString PIC X(10000)
```

Palabra clave de esquema JSON	Descripción de datos COBOL
Todo de: "type": "null" "type": [] "enum": [] "allOf" "anyOf" "noneOf" "not"	No soportada

Palabra clave de esquema JSON	Descripción de datos COBOL
"\$schema": "http://json-schema.org/draft-04/schema#"	Esta palabra clave se ignora, pero se supone que es compatible con la especificación de esquema borrador 04 JSON.
"title": "same text" "description": "more text"	No se tienen en cuenta estas palabras clave.
"format": "<predefined values>"	La palabra clave "format" se utiliza para modificar la estructura generada o el valor de tiempo de ejecución. Consulte la siguiente información de la tabla para ver el uso soportado de "format".
"type": "array", "items": {<JSON Sub-schema>}, "additionalItems": false, "maxItems": m , "minItems": n	<p>La única forma de matriz JSON soportada actualmente es un número repetido de valores del mismo tipo. El <JSON Sub-schema> debe definir un "type" soportado , pero ese "type" no puede ser "array". Se trata de una restricción sobre la estructura de lenguaje generada.</p> <p>Se supone que los "additionalItems" son falsos y no se soporta ningún otro valor.</p> <p>Si "minItems" y "maxItems" están presentes, pero no iguales, la matriz se trata como una cardinalidad fija, de lo contrario, se trata como una cardinalidad variable. Consulte “Matrices de variables de elementos en DFHJS2LS” en la página 283.</p>
"type": "array", "uniqueItems": true	"uniqueItems" no se soportan con matrices JSON.

Palabra clave de esquema JSON	Descripción de datos COBOL
<pre>"type": "object", "additionalProperties": false, "properties": { ["<element name>": {<JSON Sub-schema> } [,]]* } "required": [["<element name>" [,]]*]</pre>	<p>La única forma de objeto JSON soportado actualmente es un conjunto fijo de elementos con nombre.</p> <p>Esto generará una estructura (o subestructura) utilizando los nombres de elemento.</p> <p>Se supone que las "additionalProperties" son false, y no se soporta ningún otro valor.</p> <p>Cualquier elemento del objeto "properties" se considera "optional" si no está en la matriz "required" o si no existe la matriz "required". A un elemento "optional" se le proporciona una ordinalidad variable de cero para X; donde X es 1 o el número máximo de elementos de la matriz, donde el elemento se define como una matriz. Consulte "Matrices de variables de elementos en DFHJS2LS" en la página 283.</p>
<pre>"type": "object", "maxProperties": m, "minProperties": n, "patternProperties": {}, "dependencies":</pre>	<p>No se soporta ninguna de estas palabras claves con objetos JSON.</p>
<pre>"type": "string" "maxLength": m "pattern": "regular expression", "minLength": l</pre>	<p>PIC X(z)</p> <p>donde el valor de z se basa en m , pero depende de los valores del parámetro CHAR-VARYING.</p> <p>m se basa en la palabra clave "maxLength" y se trata como una serie de longitud fija.</p> <p>Las restricciones "pattern" y "minLength" se pasan a través de la estructura de lenguaje sólo como un comentario.</p>
<pre>"type": "string" "maxLength": m</pre>	<p>Cuando CCSID=1200 está en el nivel de correlación 4.0 y superior:</p> <p>PIC N(z) USAGE NATIONAL</p> <p>donde el valor de z se basa en m , pero depende de los valores del parámetro CHAR-VARYING.</p> <p>m se basa en la palabra clave "maxLength" y se trata como una serie de longitud fija.</p>

Palabra clave de esquema JSON	Descripción de datos COBOL
<pre>"*name*":{ "type":"string", "format":"date-time" }</pre>	<p>PIC S9(15) COMP-3</p> <p>Todo soportado cuando DATETIME=PACKED15</p> <p>Tenga en cuenta que "maxLength" y "minLength" no se soportan para este formato</p>
<pre>"*name*":{ "type":"string", "format":"uri" }</pre>	<p>PIC X(<i>m</i>)</p> <p>donde <i>m</i> se basa en la palabra clave "maxLength" y se trata como una serie de longitud fija.</p> <p>Cuando CCSID=1200 está en el nivel de correlación 4.0 y superior:</p> <p>PIC N(<i>m</i>) USAGE NATIONAL</p> <p>donde <i>m</i> se basa en la palabra clave "maxLength" y se trata como una serie de longitud fija.</p>
<pre>"*name*":{ "type":"string", "format":"base64Binary" }</pre>	<p>PIC X(<i>m</i>)</p> <p>donde <i>m</i> se basa en la palabra clave "maxLength"</p>
<pre>"*name*":{ "type":"string", "format":"hexBinary" }</pre>	<p>PIC X(<i>m</i>)</p> <p>donde <i>m</i> se basa en la palabra clave "maxLength"</p>
<pre>"*name*":{ "type":"string", "format":"<predefined>" }</pre>	<p>PIC X(<i>m</i>)</p> <p>donde <i>m</i> se basa en la palabra clave "maxLength" y se trata como una serie de longitud fija, y donde <predefined> es uno de: <i>email</i> , <i>hostname</i> , <i>ipv4</i> o <i>ipv6</i> . Se utiliza un "pattern" relevante y se pasa al comentario.</p> <p>Cuando CCSID=1200 está en el nivel de correlación 4.0 y superior:</p> <p>PIC N(<i>m</i>) USAGE NATIONAL</p> <p>donde <i>m</i> se basa en la palabra clave "maxLength" y se trata como una serie de longitud fija, y donde <predefined> es uno de: <i>email</i> , <i>hostname</i> , <i>ipv4</i> o <i>ipv6</i> . Se utiliza un "pattern" relevante y se pasa al comentario.</p>
<pre>"type":"boolean"</pre>	<p>PIC X DISPLAY</p> <p>El valor x'00' implica falso, x'01' implica verdadero.</p>

Palabra clave de esquema JSON	Descripción de datos COBOL
"type": "integer", "maxExclusive": true, "minExclusive": true, "multipleOf": n	Las restricciones "maxExclusive" y "minExclusive" se pasan a la estructura de lenguaje sólo como un comentario. "multipleOf" se ignora.
"type": "integer", minimum=0, maximum=255	Nivel de correlación 3.0 e inferior: PIC X DISPLAY Nivel de correlación 4.0 y superior (o cuando se ha especificado el parámetro INTEGER-AS-PIC9, independientemente del nivel de correlación) : PIC 9(z) COMP-5 SYNC o PIC 9(z) DISPLAY donde $10^{(z-1)} < m \leq 10^z$
"type": "integer", minimum:-128, maximum:127	Nivel de correlación 3.0 e inferior: PIC X DISPLAY Nivel de correlación 4.0 y superior (o cuando se ha especificado el parámetro INTEGER-AS-PIC9, independientemente del nivel de correlación) : PIC S9(z) COMP-5 SYNC o PIC S9(z) DISPLAY donde $10^{(z-1)} < m \leq 10^z$
"type": "integer", minimum:0, maximum; m	PIC 9(z) COMP-5 SYNC o PIC 9(z) DISPLAY donde $10^{(z-1)} < m \leq 10^z$
"type": "integer", minimum:- m , maximum: m -1	PIC S9(z) COMP-5 SYNC o PIC S9(z) DISPLAY donde $10^{(z-1)} < m \leq 10^z$

Palabra clave de esquema JSON	Descripción de datos COBOL
<pre>"type": "number", "maximum": m, "minimum": n, "maxExclusive": true, "minExclusive": true, "multipleOf": n</pre>	<p>"maximum", "minimum", "maxExclusive" y "minExclusive" se pasan a la estructura de lenguaje sólo como un comentario.</p> <p>"multipleOf" se ignora.</p>
<pre>"type": "number" "format": "decimal"</pre>	<p>PIC 9(<i>p</i>)V9(<i>n</i>) COMP-3</p> <p>donde <i>p</i> y <i>n</i> son valores predeterminados.</p>
<pre>"type": "number" "format": "float"</pre>	<p>Nivel de correlación 1.1 e inferior:</p> <ul style="list-style-type: none"> PIC X(32) <p>Nivel de correlación 1.2 y superior:</p> <ul style="list-style-type: none"> COMP-1 <p>Nota: La representación de datos de coma flotante hexadecimal (HFP) de IBM no es exactamente lo mismo que la representación IEEE-754-1985 utilizada para JSON. Algunos valores podrían no convertirse de forma exacta de una representación a la otra.</p> <p>Algunos valores sumamente grandes o pequeños podrían no ser válidos para los tipos de datos flotantes. Algunos valores podrían perder precisión cuando se convierten a o desde la representación HFP. Si las conversiones precisas son importantes, considere sustituir el uso de tipos de datos COMP-1 por alternativas de precisión fija.</p>
<pre>"type": "number" "format": "double"</pre>	<p>Nivel de correlación 1.1 e inferior:</p> <ul style="list-style-type: none"> PIC X(32) <p>Nivel de correlación 1.2 y superior:</p> <ul style="list-style-type: none"> COMP-2 <p>Nota: La representación de datos de coma flotante hexadecimal (HFP) de IBM no es exactamente lo mismo que la representación IEEE-754-1985 utilizada para JSON. Algunos valores podrían no convertirse de forma exacta de una representación a la otra.</p> <p>Algunos valores sumamente grandes o pequeños podrían no ser válidos para los tipos de datos dobles. Algunos valores podrían perder precisión cuando se convierten a o desde la representación HFP. Si las conversiones precisas son importantes, considere sustituir el uso de tipos de datos COMP-2 por alternativas de precisión fija. .</p>

Nota: CICS no puede transformar valores enteros mayores que el valor máximo para una serie larga firmada ($2^{63} - 1$) a menos que esté encerrada entre comillas.

Nota: Los valores mínimo y máximo especificados en el esquema para tipos numéricos se utilizan únicamente para correlacionar con un tipo de datos COBOL. Los datos no se validan con estos valores durante el tiempo de ejecución.

Algunos de los tipos de esquema que se muestran en la tabla se correlacionan con un formato COBOL de COMP-5 SYNC o de DISPLAY, dependiendo de los valores (si los hubiera) que se especifican en las palabras clave minimum y maximum:

- Para tipos firmados (short , int y long), se utiliza DISPLAY cuando se especifica lo siguiente:

```
"maximum":  
a  
"minimum":  
-a
```

donde *a* es una serie de '9'.

- Para los tipos sin firmar (unsignedShort , unsignedInt y unsignedLong), se utiliza DISPLAY cuando se especifica lo siguiente:

```
"maximum":  
a  
"minimum":0
```

donde *a* es una serie de '9'.

Cuando se especifica otro valor, o no se especifica ningún valor, se utiliza COMP-5 SYNC.

Correlación de C y C++ con esquemas JSON:

El programa de utilidad DFHLS2JS soporta correlaciones entre tipos de datos C y C++ y definiciones de esquema JSON.

Los nombres C y C++ se convierten en nombres JSON según las reglas siguientes:

1. Los caracteres que no son válidos en los nombres de propiedad JSON se sustituyen por ' X '.
Por ejemplo, monthly-total pasa a ser monthlyXtotal.
2. Los nombres duplicados se convierten en exclusivos añadiendo uno o varios dígitos numéricos.
Por ejemplo, dos instancias de year pasan a ser year y year1.

DFHLS2JS correlaciona tipos de datos C y C++ con elementos de esquema según la tabla siguiente. Los tipos C y C++ que no se muestran en la tabla no son compatibles con DFHLS2JS. El calificador _Packed se soporta para las estructuras. Se aplican estas restricciones:

- Los archivos de cabecera deben contener una instancia de alto nivel struct.
- No puede declarar un tipo de estructura que se contiene a sí mismo como miembro.
- Los siguientes tipos de datos C y C++ no se soportan:
decimal
long double

wchar_t (sólo C++)

- Los siguientes caracteres se ignoran si están presentes en el archivo de cabecera.

Especificadores de clase de almacenamiento:

auto
register
static
extern
mutable

Calificadores

const
volatile
_Export (sólo C++)

Especificadores de función

inline (sólo C++)
Virtual (sólo C++)

Valores iniciales

- El archivo de cabecera no debe contener estos elementos:

Uniones

Declaraciones de clase

Tipos de datos de enumeración

Variables de tipo de puntero

Declaraciones de plantilla

Las macros predefinidas; es decir, macros son nombres que comienzan y finalizan con dos caracteres de subrayado (__)

La secuencia de continuación de línea (un símbolo \ que va seguido inmediatamente por un carácter de nueva línea)

Declaradores de funciones de prototipos

Directivas de preprocesamiento

Campos de bits

La palabra clave __cdecl (o _cdecl) (sólo C++)

- El programador de aplicación debe utilizar un compilador de 32 bits para asegurar que un int se correlaciona con 4 bytes.

- No se soportan las siguientes palabras clave de C++ reservadas:

explicit
using
namespace
typename
typeid

- Si el parámetro **MAPPING-LEVEL** se establece en 1.2 o superior y el parámetro **CHAR-VARYING** se establece en NULL, las matrices de caracteres se correlacionan con una string y se procesan como series terminadas en nulo.
- Si el parámetro **MAPPING-LEVEL** se establece en 1.2 o superior y el parámetro **CHAR-VARYING** se establece en BINARY, las correlaciones de caracteres se correlacionan con xsd:base64Binary y se procesan como datos binarios.
- Si el parámetro **MAPPING-LEVEL** se establece en 1.2 o superior y el parámetro **CHAR-VARYING** se establece en COLLAPSE, <xsd:whiteSpace value="collapse"/> se genera para las series.

Tipos de datos C y C++	Esquema simpleType
char[<i>z</i>]	"type":"string" "maxLength": <i>z</i>
char16_t[<i>n</i>]	En el nivel de correlación 4.0 y superior: "type":"string" "maxLength": <i>n</i> En tiempo de ejecución, CICS rellena el campo de estructura de datos de aplicación con datos UTF-16.
char[8] Se soporta en el nivel de correlación 3.0 y superior cuando DATETIME=PACKED15	"type":"string" "format":"date-time" El formato de la indicación de fecha y hora se define por RFC3339.
char short int long long long	"type":"integer", "minimum":- (<i>n</i> + 1), "maximum": <i>n</i> donde <i>n</i> es el valor máximo que puede representarse por el primitivo.
unsigned char unsigned short unsigned int unsigned long unsigned long long	"type":"integer", "minimum":0, "maximum": <i>n</i> donde <i>n</i> es el valor máximo que puede representarse por el primitivo.
bool (sólo C++)	"type":"boolean"
float Se soporta en un nivel de correlación 1.2 y superiores.	"type":"number", "format":"float" Nota: La representación de datos de coma flotante hexadecimal (HFP) de IBM no es exactamente lo mismo que la representación IEEE-754-1985 utilizada para JSON. Algunos valores podrían no convertirse de forma exacta de una representación a la otra. Algunos valores sumamente grandes o pequeños podrían no ser válidos para los tipos de datos flotantes. Algunos valores podrían perder precisión cuando se convierten a o desde la representación HFP. Si las conversiones precisas son importantes, considere sustituir el uso de tipos de datos flotantes por alternativas de precisión fija.
double Se soporta en un nivel de correlación 1.2 y superiores.	"type":"number", "format":"double" Nota: La representación de datos de coma flotante hexadecimal (HFP) de IBM no es exactamente lo mismo que la representación IEEE-754-1985 utilizada para JSON. Algunos valores podrían no convertirse de forma exacta de una representación a la otra. Algunos valores sumamente grandes o pequeños podrían no ser válidos para los tipos de datos dobles. Algunos valores podrían perder precisión cuando se convierten a o desde la representación HFP. Si las conversiones precisas son importantes, considere sustituir el uso de tipos de datos dobles por alternativas de precisión fija.

Tipos de datos C y C++	Esquema simpleType
<i>type name</i> [<i>n</i>]	<p>Para primitivos:</p> <pre> "type": "array" "maxItems": <i>n</i> "minItems": <i>n</i> "items": { "type": "object", "properties": { <i>name</i> : { <i>type JSON</i> } } } "required": [<i>name</i>] </pre> <p>Para estructuras:</p> <pre> "type": "array" "maxItems": <i>n</i> "minItems": <i>n</i> "items": { <i>type JSON</i> } </pre> <p>donde <i>type JSON</i> es la representación del esquema JSON del tipo C o C++.</p>

Correlación de esquema JSON con C y C++:

Los programas de utilidad DFHJS2LS soportan correlaciones entre esquemas JSON y tipos de datos C y C++.

Los asistentes de CICS generan nombres de campo exclusivos y válidos para variables C y C++ de los nombres de elemento de esquema que utilizan las reglas siguientes:

1. Caracteres que no sean A-Z, a-z, 0-9, o _ se sustituyen por ' X '.
Por ejemplo, `monthly-total` pasa a ser `monthlyXtotal`.
2. Si el primer carácter no es un carácter alfabético, se sustituye por una ' X ' inicial.
Por ejemplo, `_monthlysummary` pasa a ser `Xmonthlysummary`.
3. Los nombres duplicados en el mismo ámbito se convierten en exclusivos añadiendo uno o dos dígitos numéricos a la segunda y siguientes instancias del nombre.
Por ejemplo, tres instancias de `year` pasan a ser `year` , `year1` y `year2`.
4. Un esquema JSON especifica que una variable tiene una cardinalidad variable si tiene un valor "type" de "array" , y las palabras claves "minItems" y "maxItems" se omiten o tienen diferentes valores. Si el esquema especifica que la variable tiene una cardinalidad variable, los nombres de campo se crean con los sufijos "_cont" y "_num".
Para obtener más información, consulte el apartado “Matrices de variables de elementos en DFHJS2LS” en la página 283.

5. Un esquema JSON especifica que una variable es opcional si no aparece en la matriz de palabra clave "required" que está asociada al tipo "object" del esquema JSON delimitado. Para los campos opcionales, se genera un campo adicional con un sufijo _num añadido al nombre de elemento. En el tiempo de ejecución, es cero para indicar que falta el valor de los datos JSON, y no cero si el valor está presente en los datos JSON.
6. Los nombres de campo están limitados a 50 caracteres. Si un nombre generado, incluidos prefijos y sufijos, supera esta longitud, el nombre del elemento se trunca.

DFHJS2LS correlaciona valores de tipo de esquema JSON con tipos de datos C y C++ según la tabla siguiente. También se aplican las reglas siguientes:

- Si el parámetro **MAPPING-LEVEL** se establece en 1.2 o superior y el parámetro **CHAR-VARYING** se establece en NULL, los datos de caracteres de longitud variable se correlacionan con series terminadas en nulo y se asigna un carácter adicional al terminador en nulo.
- Si el parámetro **MAPPING-LEVEL** se establece en 1.2 o superior y el parámetro **CHAR-VARYING** se establece en YES, los datos de carácter de longitud variable se correlacionan con dos elementos relacionados: un campo de longitud y un campo de datos.

Palabra clave de esquema JSON	Tipos de datos C y C++
Todo de: "type": "null" "type": [] "enum": [] "allOf" "anyOf" "noneOf" "not"	No soportado
"\$schema": "http://json-schema.org/draft-04/schema#"	Esta palabra clave se ignora, pero se supone que es compatible con la especificación de esquema borrador 04 JSON.
"title": "same text" "description": "more text"	No se tienen en cuenta estas palabras clave.
"format": "<predefined values>"	La palabra clave "format" se utiliza para modificar la estructura generada o el valor de tiempo de ejecución. Consulte la siguiente información para ver el uso soportado de "format".

Palabra clave de esquema JSON	Tipos de datos C y C++
<pre>"type": "array", "items": {<JSON Sub-schema>}, "additionalItems": false, "maxItems": m , "minItems": n</pre>	<p>La única forma de matriz JSON soportada actualmente es un número repetido de valores del mismo tipo. El <JSON Sub-schema> debe definir un "type" soportado , pero ese "type" no puede ser "array". Se trata de una restricción sobre la estructura de lenguaje generada.</p> <p>Se supone que los "additionalItems" son falsos y no se soporta ningún otro valor.</p> <p>Si "minItems" y "maxItems" están presentes, pero no iguales, la matriz se trata como una cardinalidad fija, de lo contrario, se trata como una cardinalidad variable. Consulte “Matrices de variables de elementos en DFHJS2LS” en la página 283.</p>
<pre>"type": "array", "uniqueItems": true</pre>	<p>"uniqueItems" no se soportan con matrices JSON. El <JSON Sub-schema> debe definir un "type" soportado , pero ese "type" no puede ser "array" . Se trata de una restricción sobre la estructura de lenguaje generada.</p>
<pre>"type": "object", "additionalProperties": false, "properties": { ["<element name>": {<JSON Sub-schema>} [,]]* } "required": [["<element name>" [,]]*]</pre>	<p>La única forma de objeto JSON soportado actualmente es un conjunto fijo de elementos con nombre.</p> <p>Esto generará una estructura (o subestructura) utilizando los nombres de elemento.</p> <p>Se supone que las "additionalProperties" son false, y no se soporta ningún otro valor.</p> <p>Cualquier elemento del objeto "properties" se considera "optional" si no está en la matriz "required" o si no existe la matriz "required". A un elemento "optional" se le proporciona una ordinalidad variable de cero para X; donde X es 1 o el número máximo de elementos de la matriz, donde el elemento se define como una matriz. Consulte “Matrices de variables de elementos en DFHJS2LS” en la página 283.</p>
<pre>"type": "object", "maxProperties": m, "minProperties": n, "patternProperties": {}, "dependencies":</pre>	<p>No se soporta ninguna de estas palabras claves con objetos JSON.</p>

Palabra clave de esquema JSON	Tipos de datos C y C++
<pre>"type": "string" "maxLength": m "pattern": "regular expression>", "minLength": l</pre>	<p>char[z]</p> <p>donde el valor de z se basa en <i>m</i> , pero depende de los valores del parámetro CHAR-VARYING.</p> <p><i>m</i> se basa en la palabra clave "maxLength" y se trata como una serie de longitud fija.</p> <p>Las restricciones "pattern" y "minLength" se pasan a través de la estructura de lenguaje sólo como un comentario.</p>
<pre>"type": "string" "maxLength": m</pre>	<p>Cuando CCSID=1200 está en el nivel de correlación 4.0 y superior:</p> <p>char16_t[z]</p> <p>donde el valor de z se basa en <i>m</i> , pero depende de los valores del parámetro CHAR-VARYING.</p> <p><i>m</i> se basa en la palabra clave "maxLength" y se trata como una serie de longitud fija.</p>
<pre>"*name*": { "type": "string", "format": "date-time" }</pre>	<p>char[8]</p> <p>Todo soportado cuando DATETIME=PACKED15</p>
<pre>"*name*": { "type": "string", "format": "uri" }</pre>	<p>char[m]</p> <p>donde <i>m</i> se basa en la palabra clave "maxLength" y se trata como una serie de longitud fija.</p> <p>Cuando CCSID=1200 está en el nivel de correlación 4.0 y superior:</p> <p>char16_t[m]</p> <p>donde <i>m</i> se basa en la palabra clave "maxLength" y se trata como una serie de longitud fija.</p>
<pre>"*name*": { "type": "string", "format": "base64Binary" }</pre>	<p>char[m]</p> <p>donde <i>m</i> se basa en la palabra clave "maxLength".</p>
<pre>"*name*": { "type": "string", "format": "hexBinary" }</pre>	<p>char[m]</p> <p>donde <i>m</i> se basa en la palabra clave "maxLength".</p>

Palabra clave de esquema JSON	Tipos de datos C y C++
<pre> "name*":{ "type":"string", "format":"<predefined>" } </pre>	<p>char[<i>m</i>]</p> <p>donde <i>m</i> se basa en la palabra clave "maxLength" y se trata como una serie de longitud fija, y donde <predefined> es uno de: <i>email</i> , <i>hostname</i> , <i>ipv4</i> o <i>ipv6</i>. Se utiliza un "pattern" y se pasa al comentario.</p> <p>Cuando CCSID=1200 está en el nivel de correlación 4.0 y superior:</p> <p>char16_t[<i>m</i>]</p> <p>donde <i>m</i> se basa en la palabra clave "maxLength" y se trata como una serie de longitud fija, y donde <predefined> es uno de: <i>email</i> , <i>hostname</i> , <i>ipv4</i> o <i>ipv6</i> . Se utiliza un "pattern" relevante y se pasa al comentario.</p>
"type":"boolean"	<p>bool (sólo C++)</p> <p>short (sólo C)</p>
<pre> "type": "integer", "maxExclusive": true, "minExclusive": true, "multipleOf": n </pre>	<p>Las restricciones "maxExclusive" y "minExclusive" se pasan a la estructura de lenguaje sólo como un comentario.</p> <p>"multipleOf" se ignora.</p>
<pre> "type":"integer", minimum:-128, maximum:127 </pre>	signed char
<pre> "type":"integer", minimum:0, maximum:255 </pre>	unsigned char
<pre> "type":"integer", minimum:-32768, maximum:32767 </pre>	short
<pre> "type":"integer", minimum:0, maximum:65535 </pre>	unsigned short
<pre> "type":"integer", minimum:-2147483648, maximum:2147483647 </pre>	int
<pre> "type":"integer", minimum:0, maximum:4294967295 </pre>	unsigned int
<pre> "type":"integer", minimum:-9223372036854775808, maximum:9223372036854775807 </pre>	long long
<pre> "type":"integer", minimum:0, maximum:18446744073709551615 </pre>	unsigned long long

Palabra clave de esquema JSON	Tipos de datos C y C++
<pre>"type": "number", "maximum": m, "minimum": n, "maxExclusive": true, "minExclusive": true, "multipleOf": n</pre>	<p>Las restricciones "maximum", "minimum", "maxExclusive" y "minExclusive" se pasan a la estructura de lenguaje sólo como un comentario.</p> <p>"multipleOf" se ignora.</p>
<pre>"type": "number" "format": "float"</pre>	<p>Nivel de correlación 1.1 e inferior:</p> <ul style="list-style-type: none"> char[32] <p>Nivel de correlación 1.2 y superior:</p> <ul style="list-style-type: none"> float(*) <p>Nota: La representación de datos de coma flotante hexadecimal (HFP) de IBM no es exactamente lo mismo que la representación IEEE-754-1985 utilizada para JSON. Algunos valores podrían no convertirse de forma exacta de una representación a la otra. Algunos valores sumamente grandes o pequeños podrían no ser válidos para los tipos de datos flotantes. Algunos valores podrían perder precisión cuando se convierten a o desde la representación HFP. Si las conversiones precisas son importantes, considere sustituir el uso de tipos de datos flotantes por alternativas de precisión fija.</p>
<pre>"type": "number" "format": "double"</pre>	<p>Nivel de correlación 1.0 e inferior:</p> <ul style="list-style-type: none"> char[32] <p>Nivel de correlación 1.2 y superior:</p> <ul style="list-style-type: none"> double(*) <p>Nota: La representación de datos de coma flotante hexadecimal (HFP) de IBM no es exactamente lo mismo que la representación IEEE-754-1985 utilizada para JSON. Algunos valores podrían no convertirse de forma exacta de una representación a la otra. Algunos valores sumamente grandes o pequeños podrían no ser válidos para los tipos de datos dobles. Algunos valores podrían perder precisión cuando se convierten a o desde la representación HFP. Si las conversiones precisas son importantes, considere sustituir el uso de tipos de datos dobles por alternativas de precisión fija.</p>

Nota: CICS no puede transformar valores enteros mayores que el valor máximo para una serie larga firmada ($2^{63} - 1$) a menos que esté encerrada entre comillas.

Nota: Los valores mínimo y máximo especificados en el esquema para tipos numéricos se utilizan únicamente para correlacionar con un tipo de datos C o C++. Los datos no se validan con estos valores durante el tiempo de ejecución.

Correlación de PL/I con esquemas JSON:

Los programas de utilidad DFHLS2JS y DFHLS2WS soportan correlaciones entre estructuras de datos de PL/I y definiciones de esquema JSON. Puesto que el compilador de Enterprise PL/I y los compiladores de PL/I más antiguos difieren, se admiten dos opciones de lenguaje: PLI-ENTERPRISE y PLI-OTHER.

Los nombres PL/I se convierten en nombres JSON según las reglas siguientes:

1. Los caracteres que no son válidos en los nombres de propiedad JSON se sustituyen por 'x'.
Por ejemplo, `monthly$total` pasa a ser `monthlyxtotal`.
2. Los nombres duplicados se convierten en exclusivos añadiendo uno o varios dígitos numéricos.

Por ejemplo, dos instancias de `year` pasan a ser `year` y `year1`.

DFHLS2JS correlaciona tipos de datos PL/I con elementos de esquema según la tabla siguiente. Los tipos PL/I que no se muestran en la tabla no son compatibles con DFHLS2JS. También se aplican las siguientes restricciones:

- Los elementos de datos con el atributo **COMPLEX** no se soportan.
- Los elementos de datos con atributo **FLOAT** se soportan en un nivel de correlación de 1.2 o superior. Enterprise PL/I **FLOAT IEEE** no se soporta.
- Las series DBCS puras **VARYING** y **VARYINGZ** se soportan en un nivel de correlación de 1.2 o superior.
- Los elementos de datos que se especifican con **DECIMAL**(*p* , *q*) se soportan únicamente cuando $p \geq q$.
- Los elementos de datos que se especifican con **BINARY**(*p* , *q*) se soportan únicamente cuando $q = 0$.
- Si se especifica el atributo **PRECISION** para un elemento de datos, se ignora.
- No se soportan las series **PICTURE**.
- Los elementos de datos **ORDINAL** se tratan como tipos de datos **FIXED BINARY(7)**.
- Si el parámetro **MAPPING-LEVEL** se establece en 1.2 o superior y el parámetro **CHAR-VARYING** se establece en **NULL**, las matrices de caracteres se correlacionan con una `string` y se procesan como series terminadas en nulo; esta correlación no se aplica a Enterprise PL/I.
- Si el parámetro **MAPPING-LEVEL** se establece en 1.2 o superior y el parámetro **CHAR-VARYING** se establece en **BINARY**, las correlaciones de caracteres se correlacionan con `string` y se procesan como datos binarios.
- Si el parámetro **MAPPING-LEVEL** se establece en 1.2 o superior y el parámetro **CHAR-VARYING** se establece en **COLLAPSE**; los espacios en blanco inicial y final se eliminarán y varios espacios se sustituyen por un sólo espacio.

DFHLS2JS no implementa completamente algoritmos de relleno de PL/I; por lo tanto, debe declarar bytes de relleno explícitamente en la estructura de datos. DFHLS2JS emite un mensaje si detecta que faltan los bytes de relleno. Cada estructura de nivel superior debe comenzar con un límite de palabra doble y cada byte de la estructura debe correlacionarse con el límite correcto. Tenga presente este fragmento de código:

```
3 FIELD1 FIXED BINARY(7),  
3 FIELD2 FIXED BINARY(31),  
3 FIELD3 FIXED BINARY(63);
```

En este ejemplo:

- FIELD1 tiene 1 byte de longitud y se puede alinear con cualquier límite.
- FIELD2 tiene 4 bytes de longitud y debe alinearse con un límite de palabra completa.
- FIELD3 tiene 8 bytes de longitud y debe alinearse con un límite de palabra doble.

El compilador Enterprise PL/I alinea los campos en el orden siguiente:

1. FIELD3 se alinea primero porque tiene los requisitos de límite más fuertes.
2. FIELD2 se alinea en un límite de palabra completa inmediatamente antes de FIELD3.
3. FIELD1 se alinea en un límite de byte inmediatamente antes de FIELD3.

Por último, para que toda la estructura se alinee en un límite de palabra completa, el compilador inserta tres bytes de relleno inmediatamente antes de FIELD1.

Puesto que DFHLS2JS no inserta bytes de relleno equivalentes, debe declararlos explícitamente antes de que DFHLS2JS procese la estructura. Por ejemplo:

```
3 PAD1 FIXED BINARY(7),  
3 PAD2 FIXED BINARY(7),  
3 PAD3 FIXED BINARY(7),  
3 FIELD1 FIXED BINARY(7),  
3 FIELD2 FIXED BINARY(31),  
3 FIELD3 FIXED BINARY(63);
```

De forma alternativa, puede cambiar la estructura para declarar todos los campos como no alineados y volver a compilar la aplicación que utiliza la estructura. Para obtener más información sobre los requisitos de alineación de memoria estructural de PL/I, consulte *Enterprise PL/I Language Reference*.

Descripción de datos PL/I	Definición de esquema JSON
FIXED BINARY (<i>n</i>)	<pre>"type":"integer", "minimum":- (<i>n</i> + 1), "maximum": <i>n</i></pre> <p>donde <i>n</i> es el valor máximo que puede representarse por el primitivo.</p>
UNSIGNED FIXED BINARY(<i>n</i>) Restricción: Sólo Enterprise PL/I	<pre>"type":"integer", "minimum":0, "maximum": <i>n</i></pre> <p>donde <i>n</i> es el valor máximo que puede representarse por el primitivo.</p>

Descripción de datos PL/I	Definición de esquema JSON
<p>FIXED DECIMAL(<i>n</i> , <i>m</i>)</p>	<pre>"type":"number", "format":"decimal", "multipleOf": x , "maximum": y , "minimum":- z</pre> <p>donde:</p> <p><i>x</i> es la unidad más pequeña disponible = $1 / 10^m$</p> <p><i>y</i> es el valor máximo que se puede representar mediante la combinación de <i>n</i> y <i>m</i></p> <p><i>z</i> es el valor máximo que se puede representar mediante la combinación de <i>n</i> y <i>m</i></p>
<p>FIXED DECIMAL(15)</p> <p>Se soporta en el nivel de correlación 3.0 y superior cuando DATETIME=PACKED15</p>	<pre>"type":"string", "format":"date-time"</pre> <p>El formato de la indicación de fecha y hora se define por RFC3339.</p>
<p>BIT(<i>n</i>)</p> <p>donde <i>n</i> es un múltiplo de 8. No se soportan otros valores.</p>	<pre>"type":"string" "maxLength": m</pre> <p>donde $m = n / 8$</p>
<p>CHARACTER(<i>n</i>)</p> <p>VARYING y VARYINGZ también se soportan en el nivel de correlación 1.2 y superior.</p> <p>Restricción: VARYINGZ sólo se soporta en Enterprise PL/I</p>	<pre>"type":"string" "maxLength": n</pre>
<p>GRAPHIC(<i>n</i>)</p> <p>VARYING y VARYINGZ también se soportan en el nivel de correlación 1.2 y superior.</p> <p>Restricción: VARYINGZ sólo es compatible con Enterprise PL/I</p>	<p>En un nivel de correlación de 1.0 y 1.1, donde $m = 2 * n$:</p> <pre>"type":"string" "maxLength": m</pre> <p>En un nivel de correlación 1.2 o superior:</p> <pre>"type":"string" "maxLength": n</pre>

Descripción de datos PL/I	Definición de esquema JSON
<p>WIDECHAR(<i>n</i>)</p> <p>Restricción: Sólo Enterprise PL/I</p>	<p>En un nivel de correlación de 1.0 y 1.1, donde $m = 2 * n$:</p> <pre>"type": "string" "maxLength": m</pre> <p>En un nivel de correlación 1.2 o superior:</p> <pre>"type": "string" "maxLength": n</pre> <p>En un nivel de correlación 4.0 y superior, CICS llena el campos de estructura de datos de aplicación con datos UTF-16.</p> <pre><xsd:simpleType> <xsd:restriction base="xsd:string"> <xsd:maxLength value="n"/> <xsd:whiteSpace value="preserve"/> </xsd:restriction> </xsd:simpleType></pre>
<p>ORDINAL</p> <p>Restricción: Sólo Enterprise PL/I</p>	<pre>"type": "integer", "minimum": 0, "maximum": 255</pre>
<p>BINARY FLOAT(<i>n</i>)</p> <p>donde $n \leq 21$</p> <p>Se soporta en un nivel de correlación 1.2 y superiores.</p> <p>Nota: La representación de datos de coma flotante hexadecimal (HFP) de IBM no es exactamente lo mismo que la representación IEEE-754-1985 utilizada para JSON. Algunos valores podrían no convertirse de forma exacta de una representación a la otra. Algunos valores sumamente grandes o pequeños podrían no ser válidos para los tipos de datos flotantes. Algunos valores podrían perder precisión cuando se convierten a o desde la representación HFP. Si las conversiones precisas son importantes, considere sustituir el uso de tipos de datos BINARY FLOAT por alternativas de precisión fija.</p>	<pre>"type": "number", "format": "float"</pre>

Descripción de datos PL/I	Definición de esquema JSON
<p>BINARY FLOAT(n)</p> <p>donde $21 < n \leq 53$</p> <p>No se soportan los valores superiores a 53.</p> <p>Se soporta en un nivel de correlación 1.2 y superiores.</p> <p>Nota: La representación de datos de coma flotante hexadecimal (HFP) de IBM no es exactamente lo mismo que la representación IEEE-754-1985 utilizada para JSON. Algunos valores podrían no convertirse de forma exacta de una representación a la otra. Algunos valores sumamente grandes o pequeños podrían no ser válidos para los tipos de datos flotantes. Algunos valores podrían perder precisión cuando se convierten a o desde la representación HFP. Si las conversiones precisas son importantes, considere sustituir el uso de tipos de datos BINARY FLOAT por alternativas de precisión fija.</p>	<pre>"type": "number", "format": "double"</pre>
<p>DECIMAL FLOAT(n)</p> <p>donde $n \leq 6$</p> <p>Se soporta en un nivel de correlación 1.2 y superiores.</p> <p>Nota: La representación de datos de coma flotante hexadecimal (HFP) de IBM no es exactamente lo mismo que la representación IEEE-754-1985 utilizada para JSON. Algunos valores podrían no convertirse de forma exacta de una representación a la otra. Algunos valores sumamente grandes o pequeños podrían no ser válidos para los tipos de datos flotantes. Algunos valores podrían perder precisión cuando se convierten a o desde la representación HFP. Si las conversiones precisas son importantes, considere sustituir el uso de tipos de datos DECIMAL FLOAT por alternativas de precisión fija.</p>	<pre>"type": "number", "format": "float"</pre>

Descripción de datos PL/I	Definición de esquema JSON
<p>DECIMAL FLOAT(<i>n</i>)</p> <p>donde $6 < n \leq 16$</p> <p>No se soportan los valores superiores a 16.</p> <p>Se soporta en un nivel de correlación 1.2 y superiores.</p> <p>Nota: La representación de datos de coma flotante hexadecimal (HFP) de IBM no es exactamente lo mismo que la representación IEEE-754-1985 utilizada para JSON. Algunos valores podrían no convertirse de forma exacta de una representación a la otra. Algunos valores sumamente grandes o pequeños podrían no ser válidos para los tipos de datos flotantes. Algunos valores podrían perder precisión cuando se convierten a o desde la representación HFP. Si las conversiones precisas son importantes, considere sustituir el uso de tipos de datos DECIMAL FLOAT por alternativas de precisión fija.</p>	<pre>"type": "number", "format": "double"</pre>
<p><i>name (n) descripción de datos</i></p>	<p>Para primitivos:</p> <pre>"type": "array" "maxItems": n "minItems": n "items": { "type": "object", "properties": { name : { data description JSON } } "required": [name] }</pre> <p>Para declaraciones de datos:</p> <pre>"type": "array" "maxItems": n "minItems": n "items": { data description JSON }</pre> <p>Donde <i>descripción de datos JSON</i> es la representación del esquema JSON de la descripción de datos PL/I.</p>

Esquema JSON para correlación PL/I:

El programa de utilidad DFHJS2LS soporta correlaciones entre esquemas JSON y estructuras de datos PL/I. Puesto que el compilador de Enterprise PL/I y los compiladores de PL/I más antiguos difieren, se admiten dos opciones de lenguaje: PLI-ENTERPRISE y PLI-OTHER.

Reglas para la correlación de nombres de elemento del esquema para PL/I

Los asistentes de CICS generan nombres exclusivos y válidos para variables PL/I de los nombres de elemento del esquema utilizando las reglas siguientes:

1. Caracteres que no sean A-Z, a-z, 0-9, @, #, _ o \$ se sustituyen por ' X '.

Por ejemplo, monthly-total pasa a ser monthlyXtotal.

Puede utilizar el parámetro **MAPPING-OVERRIDES** para cambiar la forma en la que se manejan otros caracteres. Por ejemplo, si establece el valor **HYPHENS-AS-UNDERSCORES**, ningún guión del esquema JSON se convierte en un guión bajo en lugar de una X. Por ejemplo, monthly-total pasa a ser monthly_total.

2. Los nombres duplicados en el mismo ámbito se convierten en exclusivos añadiendo uno o dos dígitos numéricos a la segunda y siguientes instancias del nombre.

Por ejemplo, tres instancias de year se convierten en year , year1 y year2.

3. Un esquema JSON especifica que una variable tiene una cardinalidad variable si tiene un valor "type" de "array" , y las palabras claves "minItems" y "maxItems" se omiten o tienen diferentes valores. Si el esquema especifica que la variable tiene una cardinalidad variable, los nombres de campo se crean con los sufijos "_cont" y "_num".

Para obtener más información, consulte el apartado "Matrices de variables de elementos en DFHJS2LS" en la página 283.

4. Un esquema JSON especifica que una variable es opcional si no aparece en la matriz de palabra clave "required" que está asociada al tipo "object" del esquema JSON delimitado. Para los campos opcionales, se genera un campo adicional con un sufijo _num añadido al nombre de elemento. En el tiempo de ejecución, es cero para indicar que falta el valor de los datos JSON, y no cero si el valor está presente en los datos JSON.
5. Los nombres de campo están limitados a 31 caracteres. Si un nombre generado, incluidos prefijos y sufijos, supera esta longitud, el nombre del elemento se trunca.

La longitud total del nombre resultante es de 31 caracteres o menos.

Reglas para la correlación de tipos de esquema para PL/I

DFHJS2LS correlaciona valores de tipos de esquema con tipos de datos PL/I según la tabla siguiente. También tenga en cuenta los siguientes puntos:

- Si el parámetro **MAPPING-LEVEL** se establece en 1.2 o superior y el parámetro **CHAR-VARYING** se establece en NULL, los datos de caracteres de longitud variable se correlacionan con series terminadas en nulo y se asigna un carácter adicional al terminador en nulo.
- Si el parámetro **MAPPING-LEVEL** se establece en 1.2 o superior y el parámetro **CHAR-VARYING** no se especifica, de forma predeterminada, los datos de caracteres de longitud variable se correlacionan con un tipo de datos VARYINGZ para Enterprise PL/I y el tipo de datos VARYING para Otros PL/I.

- Los datos binarios de longitud variables se correlacionan con un tipo de datos VARYING si es menor de 32 768 bytes y con un contenedor si es mayor de 32 768 bytes.

Palabra clave de esquema JSON	Descripción de datos PL/I
<p>Todo de:</p> <pre>"type": "null" "type": [] "enum": [] "allOf" "anyOf" "noneOf" "not"</pre>	No soportado
<pre>"\$schema": "http://json-schema.org/draft-04/schema#"</pre>	Esta palabra clave se ignora, pero se supone que es compatible con la especificación de esquema borrador 04 JSON.
<pre>"title": "same text" "description": "more text"</pre>	No se tienen en cuenta estas palabras clave.
<pre>"format": "<predefined values>"</pre>	La palabra clave "format" se utiliza para modificar la estructura generada o el valor de tiempo de ejecución. Consulte la siguiente información para ver el uso soportado de "format".
<pre>"type": "array", "items": {<JSON Sub-schema>}, "additionalItems": false, "maxItems": m , "minItems": n</pre>	<p>La única forma de matriz JSON soportada actualmente es un número repetido de valores del mismo tipo. El <JSON Sub-schema> debe definir un "type" soportado, pero ese "type" no puede ser "array". Se trata de una restricción sobre la estructura de lenguaje generada.</p> <p>Se supone que los "additionalItems" son falsos y no se soporta ningún otro valor.</p> <p>Si "minItems" y "maxItems" están presentes, pero no iguales, la matriz se trata como una cardinalidad fija, de lo contrario, se trata como una cardinalidad variable. Consulte "Matrices de variables de elementos en DFHJS2LS" en la página 283.</p>
<pre>"type": "array", "uniqueItems": true</pre>	"uniqueItems" no se soportan con matrices JSON. El <JSON Sub-schema> debe definir un "type" soportado, pero ese "type" no puede ser "array". Se trata de una restricción sobre la estructura de lenguaje generada.

Palabra clave de esquema JSON	Descripción de datos PL/I
<pre>"type": "object", "additionalProperties": false, "properties": { ["<element name>": {<JSON Sub-schema> } [,]]* } "required": [["<element name>" [,]]*]</pre>	<p>La única forma de objeto JSON soportado actualmente es un conjunto fijo de elementos con nombre.</p> <p>Esto generará una estructura (o subestructura) utilizando los nombres de elemento.</p> <p>Se supone que las "additionalProperties" son false, y no se soporta ningún otro valor.</p> <p>Cualquier elemento del objeto "properties" se considera "optional" si no está en la matriz "required" o si no existe la matriz "required". A un elemento "optional" se le proporciona una ordinalidad variable de cero para X; donde X es 1 o el número máximo de elementos de la matriz, donde el elemento se define como una matriz. Consulte "Matrices de variables de elementos en DFHJS2LS" en la página 283.</p>
<pre>"type": "object", "maxProperties": m, "minProperties": n, "patternProperties": {}, "dependencies":</pre>	<p>No se soporta ninguna de estas palabras claves con objetos JSON.</p>
<pre>"type": "string" "maxLength": m "pattern": "regular expression", "minLength": l</pre>	<p>char[z]</p> <p>donde el valor de z se basa en m , pero depende de los valores del parámetro CHAR-VARYING.</p> <p>m se basa en la palabra clave "maxLength" y se trata como una serie de longitud fija.</p> <p>Las restricciones "pattern" y "minLength" se pasan a través de la estructura de lenguaje sólo como un comentario.</p>
<pre>"type": "string" "maxLength": m</pre>	<p>Cuando CCSID=1200 está en el nivel de correlación 4.0 y superior:</p> <p>WIDECHAR(z)</p> <p>donde el valor de z se basa en m , pero depende de los valores del parámetro CHAR-VARYING.</p> <p>m se basa en la palabra clave "maxLength" y se trata como una serie de longitud fija.</p>

Palabra clave de esquema JSON	Descripción de datos PL/I
<pre>"*name*":{ "type":"string", "format":"date-time" }</pre>	<p>FIXED DECIMAL (15,0)</p> <p>Todo soportado cuando DATETIME=PACKED15</p>
<pre>"*name*":{ "type":"string", "format":"uri" }</pre>	<p>CHAR(<i>m</i>)</p> <p>donde <i>m</i> se basa en la palabra clave "maxLength" y se trata como una serie de longitud fija.</p> <p>Cuando CCSID=1200 está en el nivel de correlación 4.0 y superior:</p> <p>WIDECHAR(<i>m</i>)</p> <p>donde <i>m</i> se basa en la palabra clave "maxLength" y se trata como una serie de longitud fija.</p>
<pre>"*name*":{ "type":"string", "format":"base64Binary" }</pre>	<p>CHAR(<i>m</i>)</p> <p>donde <i>m</i> se basa en la palabra clave "maxLength".</p>
<pre>"*name*":{ "type":"string", "format":"hexBinary" }</pre>	<p>CHAR(<i>m</i>)</p> <p>donde <i>m</i> se basa en la palabra clave "maxLength".</p>
<pre>"*name*":{ "type":"string", "format":"<predefined>" }</pre>	<p>CHAR(<i>m</i>) donde <i>m</i> se basa en la palabra clave "maxLength" y se trata como una serie de longitud fija, y <predefined> es una de: <i>email</i> , <i>hostname</i> , <i>ipv4</i> o <i>ipv6</i>. Se pasa un "pattern" relevante al comentario.</p> <p>Cuando CCSID=1200 está en el nivel de correlación 4.0 y superior:</p> <p>WIDECHAR(<i>m</i>)</p> <p>donde <i>m</i> se basa en la palabra clave "maxLength" y se trata como una serie de longitud fija, y donde <predefined> es uno de: <i>email</i> , <i>hostname</i> , <i>ipv4</i> o <i>ipv6</i> . Se utiliza un "pattern" relevante y se pasa al comentario.</p>

Palabra clave de esquema JSON	Descripción de datos PL/I
"type": "boolean"	<p>Nivel de correlación 1.1 e inferior:</p> <p>Enterprise PL/I SIGNED FIXED BINARY (7)</p> <p>Otros PL/I FIXED BINARY (7)</p> <p>Nivel de correlación 1.2 y superior:</p> <p>Enterprise PL/I BIT(7) BIT(1)</p> <p>Otros PL/I BIT(7) BIT(1)</p> <p>donde se proporciona BIT(7) para la alineación y BIT(1) contiene el valor correlacionado booleano.</p>
"type": "integer", "maxExclusive": true, "minExclusive": true, "multipleOf": n	<p>Las restricciones "maxExclusive" y "minExclusive" se pasan a la estructura de lenguaje sólo como un comentario.</p> <p>"multipleOf" se ignora.</p>
"type": "integer", minimum: -128, maximum: 127	<p>Enterprise PL/I SIGNED FIXED BINARY (7)</p> <p>Otros PL/I FIXED BINARY (7)</p>
"type": "integer", minimum: 0, maximum: 255	<p>Enterprise PL/I UNSIGNED FIXED BINARY (8)</p> <p>Otros PL/I FIXED BINARY (8)</p>
"type": "integer", minimum: -32768, maximum: 32767	<p>Enterprise PL/I SIGNED FIXED BINARY (15)</p> <p>Otros PL/I FIXED BINARY (15)</p>
"type": "integer", minimum: 0, maximum: 65535	<p>Enterprise PL/I UNSIGNED FIXED BINARY (16)</p> <p>Otros PL/I FIXED BINARY (16)</p>
"type": "integer", minimum: -2147483648, maximum: 2147483647	<p>Enterprise PL/I SIGNED FIXED BINARY (31)</p> <p>Otros PL/I FIXED BINARY (31)</p>

Palabra clave de esquema JSON	Descripción de datos PL/I
"type": "integer", minimum: 0, maximum: 4294967295	<p>Nivel de correlación 1.1 e inferior:</p> <p>Enterprise PL/I UNSIGNED FIXED BINARY(32)</p> <p>Nivel de correlación 1.2 y superior:</p> <p>Enterprise PL/I CHAR(<i>y</i>) donde <i>y</i> es una longitud fija menor de 16 MB.</p> <p>Todos los niveles de correlación:</p> <p>Otros PL/I BIT(64)</p>
"type": "integer", minimum: -9223372036854775808, maximum: 9223372036854775807	<p>Nivel de correlación 1.1 e inferior:</p> <p>Enterprise PL/I SIGNED FIXED BINARY(63)</p> <p>Nivel de correlación 1.2 y superior:</p> <p>Enterprise PL/I CHAR(<i>y</i>) donde <i>y</i> es una longitud fija menor de 16 MB.</p> <p>Todos los niveles de correlación:</p> <p>Otros PL/I BIT(64)</p>
"type": "integer", minimum: 0, maximum: 18446744073709551615	<p>Nivel de correlación 1.1 e inferior:</p> <p>Enterprise PL/I UNSIGNED FIXED BINARY(64)</p> <p>Nivel de correlación 1.2 y superior:</p> <p>Enterprise PL/I CHAR(<i>y</i>) donde <i>y</i> es una longitud fija menor de 16 MB.</p> <p>Otros PL/I BIT(64)</p>
"type": "number" "description": "decimal"	FIXED DECIMAL(<i>n</i> , <i>m</i>)

Palabra clave de esquema JSON	Descripción de datos PL/I
<pre>"type": "number" "description": "float"</pre>	<p>Niveles de correlación 1.0 y 1.1:</p> <ul style="list-style-type: none"> CHAR(32) <p>Nivel de correlación 1.2 y superior:</p> <p>Enterprise PL/I DECIMAL FLOAT(6) HEXADEC</p> <p>Otros PL/I DECIMAL FLOAT(6)</p> <p>Nota: La representación de datos de coma flotante hexadecimal (HFP) de IBM no es exactamente lo mismo que la representación IEEE-754-1985 utilizada para JSON. Algunos valores podrían no convertirse de forma exacta de una representación a la otra. Algunos valores sumamente grandes o pequeños podrían no ser válidos para los tipos de datos flotantes. Algunos valores podrían perder precisión cuando se convierten a o desde la representación HFP. Si las conversiones precisas son importantes, considere sustituir el uso de tipos de datos DECIMAL FLOAT por alternativas de precisión fija.</p>
<pre>"type": "number" "description": "double"</pre>	<p>Niveles de correlación 1.0 y 1.1:</p> <ul style="list-style-type: none"> CHAR(32) <p>Nivel de correlación 1.2 y superior:</p> <p>Enterprise PL/I DECIMAL FLOAT(16) HEXADEC</p> <p>Otros PL/I DECIMAL FLOAT(16)</p> <p>Nota: La representación de datos de coma flotante hexadecimal (HFP) de IBM no es exactamente lo mismo que la representación IEEE-754-1985 utilizada para JSON. Algunos valores podrían no convertirse de forma exacta de una representación a la otra. Algunos valores sumamente grandes o pequeños podrían no ser válidos para los tipos de datos flotantes. Algunos valores podrían perder precisión cuando se convierten a o desde la representación HFP. Si las conversiones precisas son importantes, considere sustituir el uso de tipos de datos DECIMAL FLOAT por alternativas de precisión fija.</p>

Nota: CICS no puede transformar valores enteros mayores que el valor máximo para una serie larga firmada ($2^{63} - 1$) a menos que esté encerrada entre comillas.

Nota: Los valores mínimo y máximo especificados en el esquema para tipos numéricos se utilizan únicamente para correlacionar con un tipo de datos PL/I. Los datos no se validan con estos valores durante el tiempo de ejecución.

Matrices de variables de elementos

XML puede contener una matriz con distintos números de elementos. En general, los documentos WSDL y esquemas XML que contienen varios números de elementos no se correlacionan de forma eficiente en una única estructura de datos de lenguaje de alto nivel. CICS utiliza correlaciones basadas en contenedores o correlaciones en línea para gestionar varios elementos en XML.

Una matriz con varios elementos se representa en el esquema XML utilizando los atributos `minOccurs` y `maxOccurs` en la declaración del elemento:

- El atributo `minOccurs` especifica el número de veces mínimo que se puede dar el elemento. Puede tener un valor de 0 o un entero positivo.
- El atributo `maxOccurs` especifica el número de veces máximo que se puede dar el elemento. Puede tener un valor de un entero positivo mayor o igual al valor del atributo `minOccurs`. También puede tomar un valor de `unbounded`, que indica que no se aplica límite superior al número de veces que se puede dar el elemento.
- El valor predeterminado para ambos atributos es 1.

Este ejemplo indica una serie de 8 bytes que es opcional; es decir, que puede que no ocurra nunca o que ocurra una vez en el mensaje SOAP o XML de aplicación:

```
<xsd:element name="component"
minOccurs="0" maxOccurs="1">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:length value="8"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

El ejemplo siguiente indica una serie de 8 bytes que debe tener lugar al menos una vez:

```
<xsd:element name="component"
minOccurs="1" maxOccurs="unbounded">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:length value="8"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

En general, los documentos WSDL que contienen varios números de elementos no se correlacionan de forma eficiente en una única estructura de datos de lenguaje de alto nivel. Por lo tanto, para gestionar estos casos, CICS utiliza una serie de estructuras de datos conectadas que se pasan al programa de aplicación en una serie de contenedores. Estas estructuras se utilizan como entrada y salida de la aplicación:

- Cuando CICS transforma XML en datos de aplicación, llena estas estructuras con los datos de aplicación y la aplicación los lee.
- Cuando CICS transforma los datos de aplicación en XML, lee los datos de aplicación en las estructuras que ha estado llenando la aplicación.

El formato de estas estructuras de datos se explica mejor con una serie de ejemplos. El XML puede ser de un mensaje SOAP o de una aplicación. Estos

ejemplos utilizan una matriz de campos simples de 8 bytes. Sin embargo, el modelo soporta matrices de tipos de datos complejos y matrices de tipos de datos que contienen otras matrices.

Número fijo de elementos

El primer ejemplo ilustra un elemento que se produce tres veces exactamente:

```
<xsd:element name="component"
  minOccurs="3" maxOccurs="3">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:length value="8"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

En este ejemplo, debido a que el número de veces que se da un elemento se conoce de antemano, se puede representar como una matriz de longitud fija en una sencilla declaración COBOL (o el equivalente en otros lenguajes):

```
05 component PIC X(8) OCCURS 3 TIMES
```

Número variado de elementos en el nivel de correlación 2 e inferior

Este ejemplo ilustra un elemento obligatorio que puede encontrarse de una a cinco veces:

```
<xsd:element name="component"
  minOccurs="1" maxOccurs="5">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:length value="8"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

La estructura de datos principal contiene una declaración de dos campos. Cuando CICS transforma el XML en datos binarios, el primer campo component-num contiene el número de veces que aparece el elemento en el XML, y el segundo campo, component-cont, contiene el nombre del contenedor:

```
05 component-num PIC S9(9) COMP-5
05 component-cont PIC X(16)
```

Una segunda estructura de datos contiene la declaración del elemento:

```
01 DFHWS-component
02 component PIC X(8)
```

Debe examinar el valor de component-num (que contendrá un valor en el rango de 1 a 5) para averiguar cuántas veces se da el elemento. El contenido del elemento está en el contenedor denominado component-cont; el contenedor tiene una matriz de elementos, donde cada elemento está correlacionado por la estructura de datos DFHWS-component.

Si minOccurs="0" y maxOccurs="1", el elemento es opcional. Para procesar la estructura de datos en el programa de aplicación, debe examinar el valor de component-num:

- Si es cero, el mensaje no tiene elemento de componente y el contenido de component-cont no está definido.

- Si es uno, el elemento de componente está en el contenedor denominado `component-cont`.

El contenido del contenedor está correlacionado por la estructura de datos `DFHWS-component`.

Nota: Si el mensaje SOAP consta de un único elemento recurrente, `DFHWS2LS` genera dos estructuras de lenguaje. La estructura de lenguaje principal contiene el número de elementos en la matriz y el nombre del contenedor que tiene la matriz de elementos. La segunda estructura de lenguaje correlaciona una única instancia de elemento recurrente.

Número variado de elementos en el nivel de correlación 2.1 y superior

En el nivel de correlación 2.1 y superior, puede utilizar el parámetro **INLINE-MAXOCCURS-LIMIT** en los asistentes de CICS. El parámetro **INLINE-MAXOCCURS-LIMIT** especifica la forma en la que se manejan varios elementos. Las opciones de correlación para un número variado de elementos son correlaciones basadas en contenedores, descritas en “Número variado de elementos en el nivel de correlación 2 e inferior” en la página 350, o correlaciones en línea. El *valor* de este parámetro puede ser un entero positivo en el rango de 0 a 32767:

- El valor predeterminado de **INLINE-MAXOCCURS-LIMIT** es 1, que garantiza que los elementos opcionales están correlacionados en línea.
- Un valor de 0 para el parámetro **INLINE-MAXOCCURS-LIMIT** impide la correlación en línea.
- Si `maxOccurs` es menor o igual al valor de **INLINE-MAXOCCURS-LIMIT**, se utiliza la correlación en línea.
- Si `maxOccurs` es mayor que el valor de **INLINE-MAXOCCURS-LIMIT**, se utiliza la correlación basada en contenedores.

La correlación de un número variado de elementos en línea da como resultado la generación de una matriz, como sucede con el ejemplo de aparición fija anterior, y un contador. El campo `component-num` indica cuántas instancias del elemento están presentes y si están señaladas por una matriz. Para el ejemplos mostrado en “Número variado de elementos en el nivel de correlación 2 e inferior” en la página 350, cuando **INLINE-MAXOCCURS-LIMIT** es menor o igual a 5, la estructura de datos generada es como ésta:

```
05 component-num PIC S9(9) COMP-5 SYNC.
05 component OCCURS 5 PIC X(8).
```

El primer campo, `component-num`, es idéntico a la salida para el ejemplo de correlación basada en contenedores de la sección anterior. El segundo campo contiene una matriz de longitud 5 que es lo suficientemente larga para contener el número máximo de elementos que puede generar.

La correlación en línea difiere de la correlación basada en contenedores, que almacena el número de apariciones del elemento y el nombre del contenedor donde están colocados los datos, porque almacena todos los datos en el contenedor actual. Almacenar los datos en el contenedor actual normalmente mejorará el rendimiento y hará que la correlación en línea sea preferible.

Matrices de variables anidadas

Los esquemas XML y documentos WSDL complejos pueden contener elementos recurrentes variables, que a su vez contienen elementos recurrentes variables. En este caso, la estructura descrita se amplía más allá de los dos niveles descritos en los ejemplos.

Este ejemplo ilustra en elemento opcional denominado <component2> que se anida en un elemento obligatorio denominado <component1>, donde el elemento obligatorio puede encontrarse de una a cinco veces:

```
<xsd:element name="component1"
minOccurs="1" maxOccurs="5">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="component2"
minOccurs="0" maxOccurs="1">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:length value="8"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

La estructura de datos de nivel superior es exactamente la misma que en los ejemplos anteriores:

```
05 component1-num PIC S9(9) COMP-5
05 component1-cont PIC X(16)
```

Sin embargo, la segunda estructura de datos contiene estos elementos:

```
01 DFHWS-component1
02 component2-num PIC S9(9) COMP-5
02 component2-cont PIC X(16)
```

Una estructura de tercer nivel contiene estos elementos:

```
01 DFHWS-component2
02 component2 PIC X(8)
```

El número de apariciones del elemento más externo "component1" está en component1-num.

El contenedor nombrado en component1-cont contiene una matriz con varias instancias de la segunda estructura de datos DFHWS-component1.

Cada instancia de component2-cont nombra un contenedor diferente, cada uno de los cuales contiene la estructura de datos correlacionada por la estructura de tercer nivel DFHWS-component2.

Para ilustrar esta estructura, tenga en cuenta el fragmento de XML que coincide con el ejemplo:

```
<component1><component2>
string1
</component2></component1>
<component1><component2>
string2
</component2></component1>
<component1></component1>
```

<component1> se da tres veces. Las dos primeras contienen cada una una instancia de <component2>; la tercera instancia no.

En la estructura de datos de nivel superior, component1-num contiene un valor de 3. El contenedor nombrado en component1-cont tiene tres instancias de DFHWS-component1 :

1. En la primera, component2-num tiene un valor de 1, y el contenedor nombrado en component2-cont contiene *string1*.
2. En la segunda, component2-num tiene un valor de 1, y el contenedor nombrado en component2-cont contiene *string2*.
3. En la tercera, component2-num tiene un valor de 0, y el contenido de component2-cont no está definido.

En esta instancia, la estructura de datos completa se representa por medio de cuatro contenedores en total:

- La estructura de datos raíz en el contenedor DFHWS-DATA
- El contenedor nombrado en component1-cont
- Los dos contenedores nombrados en las dos primeras instancias de component2-cont.

Estructuras opcionales y xsd:choice

DFHWS2LS y DFHSC2LS soportan el uso de maxOccurs y minOccurs en elementos <xsd:sequence>, <xsd:choice>, y <xsd:all> sólo en el nivel de correlación 2.1 y superior, donde los atributos minOccurs y maxOccurs están configurados en minOccurs="0" y maxOccurs="1".

Los asistentes generan correlaciones que tratan estos elementos como si cada elemento hijo en ellos fuera opcional. Cuando implementa una aplicación con estos elementos, asegúrese de que la aplicación no genera combinaciones no válidas de opciones. Cada uno de los elementos tiene su propio campo count en la estructura de lenguajes generada, estos campos deben establecerse todos en "0" o todos en "1". Cualquier otra combinación de valores no es válida; excepto con elementos <xsd:choice>.

Los elementos <xsd:choice> indican que sólo se puede utilizar una de las opciones del elemento. Esto se soporta en todos los niveles de correlación. Los asistentes gestionan cada una de las opciones en un <xsd:choice> aunque es un elemento <xsd:sequence> con minOccurs="0" y maxOccurs="1". Tenga cuidado cuando implementa una aplicación utilizando el elemento <xsd:choice> para asegurarse de que la aplicación no genera combinaciones no válidas de las opciones. Cada uno de los elementos tiene su propio campo count en la estructura de lenguajes generada, exactamente uno debe estar establecido en '1' y los demás establecidos en '0'. Cualquier otra combinación de valores no es válida, excepto cuando el elemento <xsd:choice> es opcional, en cuyo caso es válido para todos los campos que esté establecido en '0'.

Soporte para valores de longitud variable y espacio en blanco

Puede personalizar la forma en que se gestionan los valores de longitud variable y el espacio en blanco utilizando valores en los asistentes de CICS y añadiendo facetas directamente en el esquema XML.

Normalmente, el asistente XML de CICS y el asistente de servicios web de CICS correlacionan series de datos con matrices de caracteres de longitud fija; estas matrices requieren relleno con espacios o nulos. La correlación de valores de

longitud variables a matrices de datos de longitud fina puede ser ineficaz y un almacenamiento de residuos. Si la longitud de datos es variable, se recomienda personalizar la manera en la que se manejan estas correlaciones.

Si está convirtiendo desde una estructura de lenguaje a un esquema XML o documento WSDL, se le recomienda especificar las facetas `whiteSpace` y `maxLength` en el esquema XML y configurar el parámetro **CHAR-VARYING-LIMIT** en los asistentes.

Si está convirtiendo desde un esquema XML o documento WSDL a una estructura de lenguaje, se recomienda establecer un valor adecuado para el parámetro **CHAR-VARYING** en los asistentes.

Nota: Los caracteres nulos ('x00') no son válidos en documentos XML. Los caracteres nulo de los datos de aplicación analizados por CICS se consideran para indicar el final de la serie y el valor se trunca. Cuando CICS genera datos de aplicación, lo hace según el valor del parámetro **CHAR-VARYING**. Por ejemplo, si se especifica la opción **CHAR-VARYING=NULL**, las series de longitud variable generadas por CICS terminan con un carácter nulo.

Correlación de valores de longitud variable desde XML a estructuras de lenguaje

Utilice facetas en el esquema XML o especifique determinados parámetros en los asistentes de CICS para personalizar la forma en la que se manejan las correlaciones entre el esquema XML o el documento WSDL y la estructura de lenguaje.

Los tipos de datos XML se pueden restringir utilizando facetas. Utilice las facetas de longitud (`length` , `maxLength` y `minLength`) y la faceta `whiteSpace` para personalizar cómo se manejan los datos de longitud variables en el XML.

length

Se utiliza para especificar si los datos son de longitud fija.

maxLength

Se utiliza para especificar la longitud máxima para el tipo de datos. Si no se establece este valor para un tipo de datos basado en series, la longitud máxima no está limitada.

minLength

Se utiliza para especificar la longitud mínima para el tipo de datos. Si no se establece este valor para un tipo de datos basado en series, la longitud mínima es 0.

whiteSpace

Se utiliza para especificar cómo se maneja el espacio en blanco alrededor de un valor de datos. Los espacios en blanco incluyen espacios, pestañas y líneas nuevas. La faceta `whiteSpace` se puede establecer en `preserve` , `replace` o `collapse` :

- Un valor de `preserve` mantiene cualquier espacio en blanco en los valores de datos.
- Un valor de `replace` significa que cualquier pestaña o línea nueva se sustituye con el número de espacios adecuados.
- Un valor de `collapse` significa que los en blanco iniciales, finales e incluidos se eliminan, y todas las pestañas, líneas nuevas y espacios consecutivos se sustituyen por caracteres de un sólo espacio.

Si la faceta `whiteSpace` no está establecida, se conserva el espacio en blanco.

Para obtener más información sobre las facetas de esquema XML, consulte el esquema de recomendación W3C *XML Schema Part 2: Datatypes Second Edition* <http://www.w3.org/TR/xmlschema-2/#facets>

Se pueden utilizar los parámetros siguientes en los asistentes de CICS, DFHSC2LS y DFHWS2LS, para alterar la forma en la que los datos de longitud variable se correlacionan desde el esquema XML a la estructura de lenguaje. Estos parámetros están disponibles en el nivel de correlación 1.2 o superior.

DEFAULT-CHAR-MAXLENGTH

Especifica la longitud de matriz predeterminada de los datos de caracteres en caracteres para correlaciones donde no está implicada la longitud en el esquema XML o el documento WSDL. El valor de este parámetro puede ser un entero positivo en el rango de 1 a 2 147 483 647.

Sin embargo, se recomienda especificar la longitud de caracteres máxima que desea que utilicen DFHSC2LS o DFHWS2LS directamente en el esquema XML o documento WSDL con la faceta `maxLength`. Especificando la longitud máxima directamente en el esquema XML o documento WSDL evita problemas asociados con tener un valor predeterminado global aplicado a todos los tipos de datos basados en series.

CHAR-VARYING-LIMIT

Especifica el tamaño máximo de los datos de caracteres de longitud variable que se correlacionan con la estructura de lenguaje. Si los datos de caracteres son mayores que el valor especificado en este parámetro, se correlacionan con un contenedor y el nombre de contenedor se utiliza en la estructura de lenguaje generada. El valor puede estar en el rango de 0 al valor predeterminado 32 767 bytes.

CHAR-VARYING

Especifica cómo se correlacionan los datos de caracteres de longitud variable. Si no especifica este parámetro, la correlación predeterminada depende del lenguaje especificado. Puede seleccionar estas opciones:

- **CHAR-VARYING=NO** especifica que los datos de caracteres de longitud variable se correlacionan como series de longitud fija.
- **CHAR-VARYING=NULL** especifica que los datos de caracteres de longitud variable se correlacionan con series terminadas en nulo.
- **CHAR-VARYING=YES** especifica que los datos de caracteres de longitud variable se correlacionan con un tipo de datos CHAR VARYING en PL/I. En lenguajes COBOL, C y C++, los datos de caracteres de longitud variable se correlacionan con una representación equivalente que consta de dos elementos relacionados: la longitud de datos y los datos.

El valor **CHAR-VARYING=YES** normalmente da como resultado el mejor rendimiento.

Correlación de valores de longitud variable desde estructuras de lenguaje a XML

Puede personalizar la forma en la que se manejan las correlaciones entre la estructura de lenguaje y el esquema XML o documento WSDL. Establezca el parámetro **CHAR-VARYING** en DFHLS2SC o DFHLS2WS, en COLLAPSE o NULL para cambiar la manera en la que se generan las matrices de caracteres.

Establezca la opción **CHAR-VARYING=NULL** que le dice a CICS que añada un carácter nulo al final de cada matriz de caracteres cuando genera XML.

Establezca la opción **CHAR-VARYING=COLLAPSE** que le dice a CICS que elimine automáticamente cualquier espacio final del final de las matrices de caracteres al generar XML. Esta opción está disponible sólo en los niveles de correlación 2.1 o superior y **CHAR-VARYING=COLLAPSE** es el valor predeterminado en el nivel de correlación 2.1 o superior para todos los lenguajes distintos a C y C++. Cuando se analiza el XML, se eliminan todos los espacios en blanco de inicio, final e incluidos.

Para obtener más información, consulte Soporte para valores de longitud variable y espacios en blanco en servicios web de CICS (nota técnica) .

Soporte para UTF-16 en datos de aplicación

Los servicios web de CICS soportan la conversión de datos de aplicación codificados con UTF-16 en XML o JSON y también de XML o JSON en datos de aplicación codificados en UTF-16. Utilice UTF-16 cuando necesite almacenar y procesar datos en varios lenguajes.

Los servicios web JSON y SOAP de CICS soportan la conversión de datos de aplicación codificados con UTF-16 en XML o JSON y también de XML o JSON en datos de aplicación codificados en UTF-16. Unicode es un esquema de codificación de anchura variable que permite a los sistemas manejar datos de forma eficaz.

UTF-16 es una codificación de ancho variable para Unicode, donde cada carácter está representado por 2 o 4 bytes. Los servicios web de CICS soportan CCSID 1200 para datos de aplicación, que es UTF-16 BE (big endian) con IBM Private Use Area. Este comportamiento es coherente con el soporte UTF-16 en todos los lenguajes soportados.

UTF-16 se soporta del nivel de correlación 4.0 en adelante. Puede personalizar cómo se convierten los datos de aplicación utilizando valores de correlación en los asistentes. Para obtener más información sobre niveles de correlación XML, consulte Mapping levels for the CICS assistants. Para obtener más información sobre niveles de correlación JSON, consulte Mapping levels for the CICS JSON assistants.

Nota: UTF-16 requiere más tiempo de proceso y menos eficiencia de almacenamiento que las codificaciones EBCDIC. Además, la mezcla de tipos de codificación provoca un proceso de tiempo de ejecución adicional.

Correlación UTF-16 de esquema XML o JSON a estructuras de lenguaje

El soporte para UTF-16 depende de cómo cree el servicio web. La correlación del esquema XML o JSON a estructuras de lenguaje, también conocida como correlación descendente, tiene las siguientes características. Si UTF-16 está habilitado, todos los campos de texto se correlacionan con campos UTF-16, mientras que los tipos de datos de visualización numérica en COBOL se correlacionan como EBCDIC. Para utilizar UTF-16, establezca el parámetro CCSID de DFHJS2LS, DFHSC2LS o DFHWS2LS en 1200.

Por ejemplo, si el siguiente fragmento de esquema XML estaba presente en WSDL:

```

<xsd:element name="myString" nillable="false">
<xsd:simpleType>
<xsd:restriction base="xsd:string">
<xsd:maxLength value="20"/>
</xsd:restriction>
</xsd:simpleType>
</xsd:element>

```

El asistente DFHWS2LS genera el campo siguiente en una estructura de lenguaje COBOL:

```

myString PIC N(
20
) USAGE NATIONAL

```

El parámetro CHAR-MULTIPLIER de los asistentes de servicios web se puede utilizar para especificar la longitud de un campo que generan los asistentes.

CHAR-MULTIPLIER

Cuando utiliza UTF-16, los únicos valores válidos para el parámetro **CHAR-MULTIPLIER** son 2 o 4 , donde 2 es el valor predeterminado.

CHAR-MULTIPLIER = 2 , donde el esquema describe una serie de maxlength x , genera PIC N(x) . El valor **CHAR-MULTIPLIER** = 2 no excluye el uso de pares de sustitución en una serie UTF-16, pero tiene repercusión en el número de caracteres que caben en el campo.

CHAR-MULTIPLIER = 4 genera PIC N(2x) . Si **CHAR-MULTIPLIER** = 4 , el valor en tiempo de ejecución se llena si la serie incluye caracteres que se pueden expresar en una única unidad de codificación.

Correlación UTF-16 de estructuras de lenguaje a esquema XML o JSON

Correlación de una estructura de lenguaje a un esquema XML o JSON, también conocida como correlación ascendente, se gestiona de forma distinta a la correlación descendente. Si se declara una serie UTF-16 en la estructura de lenguaje, CICS interpreta los datos como datos codificados en UTF-16, de lo contrario, se presupone que los datos están en codificación EBCDIC. El parámetro CCSID para DFHLS2JS, DFHLS2SC o DFHLS2WS indica la codificación de cualquier texto EBCDIC dentro de los datos de aplicación; no debe configurarse para indicar UTF-16.

Los tipos de datos que se interpretan como caracteres UTF-16 son los siguientes: PIC N (n) en COBOL, WIDECHAR(n) en PL/I y char16_t[n] en C y C++.

El parámetro CHAR-USAGE de los asistentes de servicios web se puede utilizar para especificar tipos de datos.

CHAR-USAGE

En COBOL, el tipo de datos nacional, PIC N , se puede utilizar para datos UTF-16 o DBCS. Este valor lo controla la opción del compilador NSYMBOL. Debe establecer el parámetro **CHAR-USAGE** en el asistente en el mismo valor que la opción de compilador NSYMBOL para asegurarse de que los datos se manejan adecuadamente. Se establece normalmente en CHAR-USAGE=NATIONAL cuando utiliza UTF-16.

Si desea mezclar tipos de datos nacionales que contienen datos UTF-16 y DBCS en el mismo libro de copias, puede utilizar los calificadores USAGE NATIONAL o USAGE DISPLAY-1 en campos individuales.

Nota: DFHLS2WS, DFHLS2SC y DFHLS2JS no soportan la cláusula de COBOL GROUP USAGE NATIONAL.

Creación de un proveedor de servicios web utilizando el asistente de servicios web

Puede crear una aplicación de proveedor de servicios desde una descripción de servicio web que cumple con WSDL 1.1 o WSDL 2.0, o desde una estructura de datos de lenguaje de alto nivel. El asistente de servicios web de CICS le ayuda a desplegar sus aplicaciones CICS en un valor de proveedor de servicios.

Acerca de esta tarea

Cuando utiliza el asistente para desplegar una aplicación CICS como proveedor de servicios, tiene dos opciones:

- Empiece con una descripción de servicio web y utilice el asistente para generar las estructuras de datos de lenguaje.

Utilice esta opción cuando esté implementando un proveedor de servicios que se ajuste a una descripción de servicio web existente.

- Empiece con las estructuras de datos de lenguaje y utilice el asistente para generar la descripción de servicio web.

Utilice esta opción cuando esté exponiendo un programa existente como servicio web y esté dispuesto a exponer aspectos de las interfaces de programa en la descripción de servicio web y los mensajes SOAP.

Puede exponer la descripción de servicio web asociada al proveedor de servicios utilizando un URI. Este URI tiene la misma vía de acceso que el URI asociado con WEBSERVICE con el sufijo ?wsdl agregado. Esto permite a los solicitantes de su empresa, o externos a ella, descubrir archivos WSDL asociados a proveedores de servicios.

Creación de una aplicación de proveedor de servicios a partir de una descripción de servicio web

Utilizando el asistente de servicios web de CICS, puede crear una aplicación de proveedor de servicios a partir de una descripción de servicio web que cumple con WSDL 1.1 o WSDL 2.0.

Antes de empezar

Antes de poder crear una aplicación de proveedor de servicio, se deben cumplir las siguientes condiciones:

- La descripción de servicios web debe estar en un archivo UNIX en z/OS y debe crear una interconexión de modalidad de proveedor adecuada en la región de CICS.
- Debe definir como OMVS El ID de usuario con el que se ejecuta DFHWS2LS.
- El ID de usuario debe tener permiso de lectura para z/OS UNIX y las bibliotecas PDS y permiso de escritura para los directorios especificados en los parámetros **LOGFILE** , **WSBIND** y **WSDL**.
- Debe asignar almacenamiento suficiente para el ID de usuario para que el ID ejecute Java. Puede utilizar cualquier versión soportada de Java. De forma predeterminada, DFHWS2LS utiliza la versión de Java especificada en el parámetro **JAVADIR**.

Acerca de esta tarea

Puede utilizar el asistente de servicios web para crear estructuras de lenguaje desde WSDL para la aplicación de proveedor de servicios. También puede utilizar un documento WSDL que se almacena en un repositorio IBM webSphere Service Registry and Repository (WSRR).

Procedimiento

1. Utilice el programa por lotes DFHWS2LS para generar un archivo de enlace de servicio web y una o varias estructuras de datos de lenguaje. DFHWS2LS contiene un conjunto de parámetros opcionales grande que le proporciona flexibilidad para crear el archivo de enlace y las estructuras de lenguaje que requiere su aplicación. Tenga en cuenta estas opciones cuando habilite una aplicación existente para los servicios web:
 - a. **¿Qué mecanismos utilizará CICS para pasar datos al programa de aplicación de proveedor de servicios?**
 - Puede utilizar canales y pasar datos a contenedores o utilizar un COMMAREA. Se recomiendan los canales y los contenedores. Especifíquelos con el parámetro **PGMINT**.
 - b. **¿Qué lenguaje desea generar?**
 - DFHWS2LS puede generar estructuras de datos de lenguaje COBOL, C/C++ o PL/I. Especifique el lenguaje utilizando el parámetro **LANG**.
 - c. **¿Qué nivel de correlación desea utilizar?**
 - Cuánto mayor sea el nivel de correlación, mayor el control y soporte que tiene disponible para el manejo de caracteres y datos binarios en tiempo de ejecución. Algunos parámetros opcionales sólo están disponibles en niveles de correlación superiores. Se recomienda utilizar el nivel de correlación más alto disponible. Especifique el nivel de correlación con el parámetro **MAPPING-LEVEL**.
 - d. **¿Qué URI desea que utilice el solicitante de servicio web?**
 - Especifique un URI relativo utilizando el parámetro **URI**; por ejemplo, **URI=/my/test/webservice**. CICS utiliza el valor cuando crea el recurso **URIMAP**.
 - e. **¿En qué transacción e ID de usuario ejecutará la solicitud y respuesta del servicio web?**
 - Puede utilizar una transacción de alias para ejecutar la aplicación para componer una respuesta para el solicitante de servicio. La transacción de alias se conecta con el ID de usuario.
 - Especifíquelo con los parámetros **TRANSACTION** y **USERID**. Estos valores se utilizan cuando se crea el recurso **URIMAP**. Si no desea utilizar una transacción específica, no utilice estos parámetros.
 - f. **¿Dónde está almacenado el documento WSDL?**
 - Si desea recuperar un documento WSDL de un servidor WSRR, en lugar de desde el sistema de archivos local, debe especificar determinados parámetros en DFHWS2LS.
 - Como mínimo, debe especificar el parámetro **WSRR-SERVER** con la ubicación del servidor WSRR y el parámetro **WSRR-NAME** con el nombre del documento WSDL que desea recuperar desde WSRR.
 - Para obtener información sobre otros parámetros que desee especificar si está utilizando WSRR, consulte “DFHWS2LS: Conversión de WSDL a lenguaje de alto nivel” en la página 506.

- g. Si desea recuperar el documento WSDL de un servidor WSRR, ¿desea hacerlo utilizando una conexión segura?
 - Puede utilizar el cifrado de capa de sockets seguros (SSL) estableciendo los parámetros adecuados para interoperar de forma segura con WSRR. Por ejemplo, consulte “Ejemplo de cómo utilizar SSL con el asistente de servicios web y WSRR” en la página 53.
 - Cuando envía DFHWS2LS, CICS genera el archivo de enlace de servicio web y lo coloca en la ubicación que especifica con el parámetro **WSBIND**. Las estructuras de lenguaje se colocan en el conjunto de datos particionados que especifica con el parámetro **PDSLIB**.
2. Copie el archivo de enlace de servicio web generado en el directorio pickup del recurso PIPELINE de modalidad de proveedor que desea utilizar para la aplicación de servicio web. Debe copiar el archivo de enlace en modalidad binaria.
3. Opcional: Copie la descripción de servicio web o el archivo de archivado que contiene una o varias descripciones de servicio web al mismo directorio que el del archivo de enlace de servicio web. El archivo de archivado debe ser un archivo .zip y el nombre de archivo debe coincidir con el nombre de archivo WSDL. Con esta copia, podrá descubrir el WSDL.
4. Escriba un programa de aplicación de proveedor de servicios para la interfaz con las estructuras de lenguaje generadas e implemente la lógica empresarial necesaria.
5. Cree el recurso WEBSERVICE y dos recursos URIMAP.
 - El recurso WEBSERVICE encapsula el archivo de enlace de servicio web en CICS y se utiliza en tiempo de ejecución.
 - El primer recurso URIMAP proporciona a CICS información para asociar el recurso WEBSERVICE a un URI específico.
 - El segundo recurso URIMAP proporciona a CICS la información para asociar el archivo de archivado WSDL o el documento WSDL a un URI específico.
 - Este URI tiene la misma vía de acceso que el URI asociado con WEBSERVICE con el sufijo ?wsdl agregado.
 - Este recurso URIMAP se crea para que los solicitantes externos puedan utilizar el URI para descubrir el archivo de archivado WSDL o el documento WSDL.
 - Este recurso URIMAP se crea sólo si la descripción de servicio web o el archivo de archivado que contienen una o más descripciones de servicio web se han copiado al mismo directorio que el del archivo de enlace de servicio web.
 - Si el directorio de recogida contiene un archivo de archivado WSDL y un documento WSDL, el URI solo devuelve el WSDL en el archivo de archivado.
 - Esta función sólo está disponible para los servicios web instalados utilizando la operación de exploración de interconexión.

Puede crear los recursos de las siguientes maneras

- a. Utilizando el mandato **PIPELINE SCAN** para crear dinámicamente el recurso WEBSERVICE y los recursos URIMAP.
- b. Defina los recursos usted mismo. Si utiliza CICS Explorer para definir un recurso WEBSERVICE en un paquete CICS, puede elegir importar un archivo de enlace de servicio web y un documento WSDL o archivo de archivado WSDL e incluirlos en el paquete. También puede generar definiciones URIMAP para soportar el servicio web y empaquetarlas en un

paquete. Para obtener más ayuda utilizando CICS Explorer para crear y editar recursos en paquetes CICS, consulte *Working with bundles in the CICS Explorer product documentation*.

Resultados

Si tiene problemas a la hora de enviar DFHWS2LS, o los recursos no se instalan correctamente, consulte *Diagnosing deployment errors*.

Creación de una aplicación de proveedor de servicios a partir de una estructura de datos

Utilizando el asistente de servicios web de CICS, puede crear una aplicación de proveedor de servicios a partir de una estructura de datos de lenguaje de alto nivel.

Antes de empezar

Antes de crear una aplicación de proveedor de servicios, asegúrese de que se completan estas condiciones previas:

- Las estructuras de datos de lenguaje de alto nivel deben cumplir los siguientes criterios:
 - Las estructuras de datos deben definirse por separado desde el programa de origen; por ejemplo, en un libro de copias COBOL.
 - Si el programa de aplicación PL/I o COBOL utiliza diferentes estructuras de datos para la entrada y la salida, las estructuras de datos deben estar definidas en dos miembros diferentes en un conjunto de datos particionados. Si se utiliza la misma estructura para la entrada y salida, la estructura debe estar definida en un único miembro.
Para C y C++, las estructuras de datos pueden estar en el mismo miembro en un conjunto de datos particionados.
- Las estructuras de datos que procesa dependen de si está utilizando un programa derivador:
 - Si está utilizando un programa derivador, el libro de copias es la interfaz para el programa derivador.
 - Si no está utilizando un programa derivador, el libro de copias es la interfaz para la lógica empresarial.
- Las estructuras de lenguaje deben estar disponibles en un conjunto de datos particionados y debe crear un recurso PIPELINE adecuado en la región de CICS:
 - Debe definir para OMVS el ID de usuario bajo el que se ejecuta DFHLS2WS.
 - El ID de usuario debe tener permiso de lectura para z/OS UNIX y las bibliotecas PDS y permiso de escritura para los directorios especificados en los parámetros **LOGFILE** , **WSBIND** y **WSDL**.
 - El ID de usuario debe tener una asignación de almacenamiento lo suficientemente grande para ejecutar Java. Puede utilizar cualquier versión soportada de Java. De forma predeterminada, DFHLS2WS utiliza la versión de Java especificada en el parámetro **JAVADIR**.

Procedimiento

Siga estos pasos para crear una aplicación de proveedor de servicios desde una estructura de datos:

1. Si la interfaz de la aplicación de proveedor de servicios utiliza canales y muchos contenedores, cree un documento de descripción de canal que describe la interfaz en XML. Debe colocar el documento de descripción de canal en un directorio adecuado en z/OS UNIX. CICS utiliza este documento para construir y deconstruir un mensaje SOAP desde los contenedores en un canal. De lo contrario, puede utilizar un contenedor en un canal y no crear un documento de descripción de canal.
 - Para obtener más información sobre cómo crear un documento de descripción de canal, consulte “Creación de un documento de descripción de canal” en la página 582.
2. Utilice el programa por lotes DFHLS2WS para generar un archivo de enlace de servicio web y una descripción de servicio web desde la estructura de lenguaje. DFHLS2WS contiene un conjunto de parámetros opcionales grande que le proporciona flexibilidad para crear el archivo de enlace y las estructuras de lenguaje que requiere su aplicación. Tenga en cuenta estas opciones cuando el servicio web esté habilitando una aplicación existente:
 - a. **¿Qué mecanismos utilizará CICS para pasar datos al programa de aplicación de proveedor de servicios?**
 - Puede utilizar canales y pasar datos a contenedores o utilizar un COMMAREA. Especifique el mecanismo utilizando el parámetro **PGMINT**. Si la interfaz de aplicación utiliza canales y muchos contenedores, especifique el parámetro **REQUEST-CHANNEL** y, opcionalmente, **RESPONSE-CHANNEL**. Sólo puede utilizar estos parámetros cuando el nivel de correlación sea 3.0 o superior.
 - b. **¿Qué nivel de descripción de servicio web (documento WSDL) desea generar?**
 - CICS genera descripciones que cumplen con los documentos WSDL 1.1 o WSDL 2.0. Si desea que la aplicación de proveedor de servicios soporte solicitudes que cumple con ambos niveles de WSDL, especifique valores para los parámetros **WSDL_1.1** y **WSDL_2.0**. Asegúrese de que los nombres de archivo son diferentes cuando utiliza más de un parámetro WSDL. Esta especificación genera dos descripciones de servicio web y un archivo de enlace.
 - c. **¿Qué versión del protocolo SOAP desea utilizar?**
 - Puede especificar la versión con el parámetro **SOAPVER**. Se recomienda que utilice el valor ALL, que proporciona flexibilidad para utilizar SOAP 1.1 o SOAP 1.2 como enlace para la descripción de servicio web en una interconexión configurada con el manejador de mensajes SOAP 1.2. Puede utilizar este parámetro sólo cuando **MINIMUM-RUNTIME-LEVEL** es 2.0 o superior.
 - d. **¿Qué nivel de correlación desea utilizar?**
 - Cuánto mayor sea el nivel de correlación, mayor el control y soporte que tiene disponible para el manejo de caracteres y datos binarios en tiempo de ejecución. Algunos parámetros opcionales sólo están disponibles en niveles de correlación superiores. Se recomienda especificar el nivel de correlación más alto disponible en el parámetro **MAPPING-LEVEL**.
 - e. **¿Qué URI desea que utilice el solicitante de servicio web?**
 - Especifique un URI absoluto utilizando el parámetro **URI**; por ejemplo **URI = http://www.example.org:80/my/test/webservice**. La parte relativa de esta dirección, /my/test/webservice, se utiliza al crear el recurso URIMAP. El URI completo se utiliza como elemento <soap:address> en la descripción de servicio web. Este uso es verdadero para los URI de HTTP y WebSphere MQ.

- f. **¿Desea publicar el documento WSDL en un IBM WebSphere Service Registry and Repository (WSRR)?**
 - Si desea publicar el documento WSDL en un WSRR, debe especificar el parámetro **WSRR-SERVER** en DFHLS2WS. Para obtener más información sobre los parámetros que especifica cuando utiliza WSRR, consulte “DFHLS2WS: Conversión de lenguaje de alto nivel a WSDL” en la página 490.
- g. **¿Si desea publicar el documento WSDL en un servidor WSRR, desea hacerlo utilizando una conexión segura?**
 - Puede utilizar el cifrado de capa de sockets seguros (SSL) estableciendo los parámetros adecuados para interoperar de forma segura con WSRR. Por ejemplo, consulte Example of how to use SSL with the web services assistant and WSRR.
 - Cuando envía DFHLS2WS, CICS genera el archivo de enlace de servicio web y lo coloca en la ubicación que especifica con el parámetro **WSBIND**. La descripción de servicio web generada se coloca en la ubicación que ha especificado con el parámetro **WSDL**, **WSDL_1.1**, o **WSDL_2.0**.
 - Si ha utilizado los parámetros WSRR en DFHLS2WS, el documento WSDL se publica en el servidor WSRR que ha especificado.
3. Revise la descripción de servicio web generada y realice cualquier personalización necesaria. Para obtener más información, consulte el apartado “Personalización de los documentos de descripción de servicios web generados” en la página 584.
4. Copie el archivo de enlace de servicio en el directorio de recogida de la interconexión de modalidad de proveedor que desea utilizar para la aplicación de servicio web. Debe copiar el archivo de enlace de servicio web en modalidad binaria.
5. Opcional: Copie la descripción de servicio web o el archivo de archivado que contiene una o varias descripciones de servicio web al mismo directorio que el del archivo de enlace de servicio web. El archivo de archivado debe ser un archivo .zip y el nombre de archivo debe coincidir con el nombre de archivo WSDL. Con esta copia, podrá descubrir el WSDL.
6. Utilice el mandato **PIPELINE SCAN** para crear dinámicamente el recurso WEBSERVICE y dos recursos URIMAP. El recurso WEBSERVICE encapsula el archivo de enlace de servicio web en CICS y se utiliza en tiempo de ejecución.
 - El primer recurso URIMAP proporciona a CICS información para asociar el recurso WEBSERVICE a un URI específico.
 - El segundo recurso URIMAP proporciona a CICS la información para asociar el archivo de archivado WSDL o el documento WSDL a un URI específico.
 - Este URI tiene la misma vía de acceso que el URI asociado con WEBSERVICE con el sufijo ?wsdl agregado.
 - Este recurso URIMAP se crea para que los solicitantes externos puedan utilizar el URI para descubrir el archivo de archivado WSDL o el documento WSDL.
 - Este recurso URIMAP se crea sólo si la descripción de servicio web o el archivo de archivado que contienen una o más descripciones de servicio web se han copiado al mismo directorio que el del archivo de enlace de servicio web.
 - Si el directorio de recogida contiene un archivo de archivado WSDL y un documento WSDL, el URI solo devuelve el WSDL en el archivo de archivado.

- Esta función sólo está disponible para los servicios web instalados utilizando la operación de exploración de interconexión.

Como alternativa, puede definir los recursos, aunque no es lo recomendable.

Resultados

Cuando haya creado correctamente los recursos CICS, la creación de la aplicación de proveedor de servicios está completa.

Si tiene problemas a la hora de enviar DFHLS2WS, o los recursos no se instalan correctamente, consulte Diagnosing deployment errors.

Qué hacer a continuación

Ponga la descripción de servicios web a disposición de todo el que necesite desarrollar un solicitante de servicio web que desee acceder al servicio.

Creación de un documento de descripción de canal

Cree un documento de descripción de canal cuando la aplicación de proveedor de servicios utiliza una interfaz de canal con muchos contenedores.

Acerca de esta tarea

Utilice un editor XML para crear un documento de descripción de canal. El esquema para la descripción de canal se denomina `channel.xsd` y está en el directorio `/usr/lpp/cicsts/cicsts52/schemas/channel` (donde `/usr/lpp/cicsts/cicsts52` es el directorio de instalación predeterminado para los archivos CICS).

Procedimiento

1. Cree un documento XML con un elemento `<channel>` y el espacio de nombres de canal de CICS:

```
<channel name="myChannel"
xmlns="http://www.ibm.com/xmlns/prod/CICS/channel">
</channel>
```

2. Añada un elemento `<container>` a cada contenedor que utiliza el programa de aplicación en el canal. Debe utilizar los atributos de nombre, tipo y uso para describir cada contenedor. El ejemplo siguiente muestra seis contenedores con diferentes valores de atributo:

```
<container name="cont1" type="char" use="required"/>
<container name="cont2" type="char" use="optional"/>
<container name="cont3" type="bit" use="required"/>
<container name="cont4" type="bit" use="optional"/>
<container name="cont5" type="bit" use="required">
<structure location="//HLQ.PDSNAME(MEMBER)"/>
</container>
<container name="cont6" type="bit" use="optional">
<structure location="//HLQ.PDSNAME(MEMBER2)"/>
</container>
```

El elemento de estructura indica que el contenido se define en una estructura de lenguaje ubicada en un miembro de conjunto de datos particionados.

3. Guarde el documento XML en z/OS UNIX.

Esquema de canal

El documento de descripción de canal debe ajustarse al siguiente esquema:

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.ibm.com/xmlns/prod/CICS/channel"
xmlns:tns="http://www.ibm.com/xmlns/prod/CICS/channel"
elementFormDefault="qualified">
  <element name="channel">
    1
    <complexType>
      <sequence>
        <element name="container" maxOccurs="unbounded" "unbounded" minOccurs="0">
          2
          <complexType>
            <sequence>
              <element name="structure" minOccurs="0">
                3
                <complexType>
                  <attribute name="location" type="string" use="required"/>
                  <attribute name="structure" type="string" use="optional"/>
                </complexType>
              </element>
            </sequence>
            <attribute name="name" type="tns:name16Type" use="required"/>
            <attribute name="type" type="tns:typeType" use="required"/>
            <attribute name="use" type="tns:useType" use="required"/>
          </complexType>
        </element>
      </sequence>
      <attribute name="name" type="tns:name16Type" use="optional" />
    </complexType>
  </element>
  <simpleType name="name16Type">
    <restriction base="string">
      <maxLength value="16"/>
    </restriction>
  </simpleType>
  <simpleType name="typeType">
    <restriction base="string">
      <enumeration value="char"/>
      <enumeration value="bit"/>
    </restriction>
  </simpleType>
  <simpleType name="useType">
    <restriction base="string">
      <enumeration value="required"/>
      <enumeration value="optional"/>
    </restriction>
  </simpleType>
</schema>
```

1. Este elemento representa un canal de CICS.
2. Este elemento representa un contenedor de CICS dentro del canal.
3. Una estructura solo se puede utilizar con contenedores en modo 'bits'. El atributo 'ubicación' indica la ubicación de un archivo que correlaciona el contenido del contenedor. El atributo 'estructura' puede utilizarse en C y C++ para indicar el nombre de la estructura.

Qué hacer a continuación

Ejecute DFHLS2WS para crear correlaciones y un documento WSDL para la aplicación de proveedor de servicios web. DFHLS2WS sitúa las correlaciones para el canal en el documento WSDL en el orden en el que se especifican los contenedores en el documento de descripción de canal.

Personalización de los documentos de descripción de servicios web generados

Los documentos de descripción de servicio web (WSDL) que genera DFHLS2WS tienen contenido generado automáticamente que puede ser adecuado que cambie antes de publicar. La personalización de documentos WSDL puede dar como resultado la regeneración del archivo de enlace de servicios web y, en algunos casos, la escritura de un programa derivador.

Acerca de esta tarea

Siga estos pasos para personalizar los documentos de descripción de servicio web generados:

Procedimiento

1. Si desea anunciar soporte para HTTPS o comunicarse utilizando WebSphere MQ, utilice el parámetro **URI** en DFHLS2WS para establecer un URI absoluto. Si no ha utilizado el parámetro **URI**, debe cambiar los elementos `<wsdl:service>` y `<wsdl:binding>` al final del documento WSDL. El WSDL generado incluye comentarios para ayudarle a hacer esos cambios. El cambio de estos elementos no requiere que vuelva a generar el archivo de enlace de servicios web.
2. Si desea proporcionar la ubicación de red del servicio web, utilice el parámetro **URI** en DFHLS2WS para establecer un URI absoluto. Si no ha utilizado el parámetro **URI**, añada los detalles a `soap:address` en el elemento `wsdl:service`.
 - a. Si está utilizando el protocolo basado en HTTP, sustituya *my-server* por el nombre de host TCP/IP de la región de CICS y *my-port* por el número de puerto del recurso TCPIP SERVICE.
 - b. Si está utilizando WebSphere MQ como protocolo de transporte, sustituya *myQueue* por el nombre de cola adecuado.

Puede realizar estos cambios sin necesitar ningún cambio para el archivo de enlace de servicios web.

Si está cambiando el nombre de puerto y el espacio de nombres sin regenerar el archivo WSBIND, la información de supervisión puede ser incorrecta en el nivel de tiempo de ejecución 2.1 y en adelante.

3. Piense en si los nombres generados automáticamente en el documento WSDL son adecuados para su fines. Puede cambiar el nombre de estos valores:
 - El `targetNamespace` del documento WSDL
 - El `targetNamespace` de los esquemas XML dentro del documento WSDL
 - El nombre `<wsdl:portType>`
 - El nombre `<wsdl:operation>`
 - El nombre `<wsdl:binding>`
 - El nombre `<wsdl:service>`
 - Los nombres de los campos en los esquemas XML en el documento WSDL.

Estos valores forman parte de la interfaz de programación a la que codifica un programa cliente. Si los nombres generados no son lo suficientemente significativos, el mantenimiento del código de aplicación puede ser más difícil durante un largo periodo de tiempo. Utilice los parámetros DFHLS2WS **REQUEST-NAMESPACE** y **RESPONSE-NAMESPACE** para cambiar el `targetNamespace` de los esquemas XML, y el parámetro **WSDL-NAMESPACE** para cambiar el `targetNamespace` del documento WSDL.

Si cambia cualquiera de estos valores, debe utilizar DFHWS2LS para volver a generar el archivo de enlace de servicios web. Las estructuras de lenguaje que

se generan no serán las mismas que las estructuras de lenguaje existentes, pero son compatibles con la aplicación existente, así que no son necesarios cambios de aplicación. Sin embargo, puede ignorar las nuevas estructuras de lenguaje y utilizar el nuevo archivo de enlace de servicios web con las estructuras originales.

4. Piense si los campos COMMAREA expuestos en esquemas XML son adecuados. Puede pensar en eliminar los campos que no son útiles para el desarrollador de clientes de servicios web:
 - Los campos que se utilizan únicamente para valores de salida se pueden eliminar del esquema que se correlaciona con las estructuras de datos de entrada.
 - Campos de relleno.
 - Anotaciones generadas de forma automática.

Si realiza cualquiera de estos cambios, debe volver a generar el archivo de enlace de servicios web utilizando DFHWS2LS. Las nuevas estructuras de lenguaje generadas no son compatibles con las estructuras de lenguaje originales, por lo que debe escribir un programa derivador para correlacionar datos desde la nueva representación a la antigua. Este programa derivador debe ejecutar un mandato **EXEC CICS LINK** para el programa de aplicación de destino y, después, correlacionar los datos devueltos.

Este nivel de personalización requiere el máximo esfuerzo, pero da como resultado las interfaces de programación más significativas para los desarrolladores de clientes de servicios web.

5. Si desea poner el documento WSDL generado a través de DFHWS2LS para crear nuevas estructuras de lenguaje, decida si mantiene las anotaciones en el documento WSDL. Las anotaciones sustituyen las reglas de correlación normales cuando DFHWS2LS genera las estructuras de lenguaje. Cuando sustituya las reglas de correlación, asegúrese de que las estructuras de lenguaje generadas son compatibles con la versión que ha utilizado DFHLS2WS. Si desea utilizar las reglas de correlación predeterminadas para generar estructuras de lenguaje, elimine las anotaciones.

Resultados

Si desea publicar el documento WSDL personalizado para un servidor IBM WebSphere Service Registry and Repository (WSRR), debe publicarlo manualmente utilizando la interfaz WSRR. Puede encontrar más información sobre WSRR en la siguiente ubicación: WebSphere Service Registry and Repository.

Ejemplo

Para obtener un ejemplo de un documento WSDL, consulte *An example of the generated WSDL document*.

Envío de un error SOAP

En un proveedor de servicios, puede utilizar la API de CICS para enviar un error SOAP a un solicitante de servicio web. El error lo puede emitir la aplicación de proveedor de servicios o el programa de proceso de cabeceras en la interconexión.

Antes de empezar

Para utilizar la API, la aplicación de proveedor de servicio debe utilizar canales y contenedores. Si la aplicación utiliza COMMAREA, escriba un programa derivador

que utiliza canales y contenedores para crear el mensaje de error SOAP. Puede utilizar la API en el programa de proceso de cabeceras sólo si se invoca directamente desde un manejador de mensajes SOAP proporcionado por CICS.

Acerca de esta tarea

Es posible que desee emitir un error SOAP para el solicitante de servicio web si la lógica de aplicación no puede satisfacer la solicitud, por ejemplo, o si hay un problema subyacente con el mensaje de solicitud. Tenga en cuenta que CICS no considera la emisión de un error SOAP como un error, por lo tanto, tiene lugar un proceso de interconexión de respuesta al mensaje normal en lugar de un proceso de error. Si desea revertir cualquier transacción, debe utilizar el programa de aplicación.

Procedimiento

1. En el programa, utilice el mandato **EXEC CICS SOAPFAULT CREATE** para enviar un error SOAP.
2. Añada la opción **CLIENT** o **SERVER** en el mandato. Esta opción indica dónde se ha producido el problema, en el lado del cliente o del servidor.
 - **CLIENT** indica que el problema es con el mensaje de solicitud que se ha recibido.
 - **SERVER** indica que el problema tiene lugar cuando CICS ha procesado el mensaje de solicitud. Este problema puede estar en un programa de aplicación, por ejemplo, es posible que no sea capaz de cumplir con la solicitud, o que sea un problema subyacente que se produce durante el proceso de interconexión.
3. Añada la opción **FAULTSTRING** y su longitud en la opción **FAULTSTRLEN** para proporcionar un resumen de porqué el proveedor de servicios ha emitido el error. El contenido de esta opción está en XML. Los datos proporcionados por la aplicación deben estar en un formato adecuado para la inclusión directa en un documento XML. Es posible que la aplicación tenga que especificar algunos caracteres como entidades XML. Por ejemplo, si el carácter `<` se utiliza en otros lugar que no sea el inicio de una etiqueta XML, la aplicación debe cambiarlo a `<`. El ejemplo siguiente muestra una opción **FAULTSTRING** incorrecta:

```
dc1 msg_faultString char(*) constant('Error: Value A
< Value B');
```

La forma correcta de especificar esta opción **FAULTSTRING** es la siguiente:

```
dc1 msg_faultString char(*) constant('Error: Value A
&lt; Value B');
```

Consejo: Para evitar el uso de entidades XML, puede derivar los datos en una construcción XML **CDATA**. Los procesadores XML no analizan datos de caracteres en esta construcción. Utilizando este método, puede especificar la siguiente opción **FAULTSTRING**:

```
dc1 msg_faultString char(*)
constant('<![CDATA[Error: Value A < Value B]]>');
```

4. Codifique la opción **DETAIL** y su longitud en la opción **DETAILLENGTH** para proporcionar los detalles de porqué el proveedor de servicios ha emitido el error. El contenido de esta opción está en XML. La misma directriz se aplica a la opción **DETAIL** que a la opción **FAULTSTRING**.
5. Opcional: Si invoca la API desde un programa de proceso de cabeceras, defina el programa en el archivo de configuración de interconexión. El programa de

proceso de cabeceras se define en el elemento `<cics_soap_1.1_handler>` ,
`<cics_soap_1.2_handler>` , `<cics_soap_1.1_handler_java>` o
`<cics_soap_1.2_handler_java>` .

Resultados

Cuando el programa emite este mandato, CICS crea el mensaje de respuesta de error SOAP en el nivel SOAP adecuado. Si la aplicación de proveedor de servicios emite el mandato, no necesita crear una respuesta SOAP y la coloca en el contenedor DFHRESPONSE. La interconexión procesa el error SOAP a través de los manejadores de mensaje y envía la respuesta al proveedor de servicios web.

Ejemplo

El mandato **SOAPFAULT CREATE** tiene varias opciones para proporcionarle flexibilidad para responder adecuadamente a un solicitante de servicio web. Aquí, hay un sencillo ejemplo de mandato completado que crea un error SOAP que se puede utilizar para SOAP 1.1 y SOAP 1.2:

```
EXEC CICS SOAPFAULT CREATE CLIENT
DETAIL(
  msg_detail
)
DETAILLENGTH(length(
  msg_detail
))
FAULTSTRING(
  msg_faultString
)
FAULTSTRLEN(length(
  msg_faultString
));
```

Puede codificar *msg_detail* y *msg_faultString* con los siguientes valores:

```
dc1 msg_detail char(*)
  constant('<ati:ExampleFault xmlns="http://www.example.org/faults"
  xmlns:ati="http://www.example.org/faults">Detailed error message
  goes here.</ati:ExampleFault>');
dc1 msg_faultString char(*) constant('Something went wrong');
```

Creación de un solicitante de servicio web utilizando el asistente de servicios web

Puede crear una aplicación de solicitante de servicio desde una descripción de servicio que cumple con WSDL 1.1 o WSDL 2.0. El asistente de CICS web le ayuda a desplegar las aplicaciones CICS en una configuración del solicitante de servicio.

Antes de empezar

La descripción de servicios web debe estar en un archivo en z/OS UNIX o debe publicarse en un servidor IBM WebSphere Services Registry and Repository (WSRR) y se debe instalar una interconexión de modalidad de solicitante en la región de CICS.

Debe asignar almacenamiento suficiente para el ID de usuario para que ese ID pueda ejecutar Java. Puede utilizar cualquier versión soportada de Java. De forma predeterminada, DFHWS2LS utiliza la versión de Java especificada en el parámetro **JAVADIR**.

Acerca de esta tarea

Cuando utiliza el asistente de servicios web de CICS para desplegar una aplicación CICS como solicitante de servicio, debe comenzar con una descripción de servicio web y generar las estructuras de datos de lenguaje a partir de ella.

Procedimiento

1. Utilice el programa por lotes DFHWS2LS para generar un archivo de enlace de servicio web y una o varias estructuras de lenguaje. Tenga en cuenta estas opciones a la hora de crear una aplicación de solicitante de servicio desde una descripción de servicio web:
 - ¿Qué nivel de correlación desea utilizar? Cuánto mayor sea el nivel de correlación, mayor el control y soporte que tiene disponible para el manejo de caracteres y datos binarios en tiempo de ejecución. Algunos parámetros opcionales sólo están disponibles en niveles de correlación superiores. Se recomienda utilizar el nivel de correlación más alto disponible.
 - ¿Qué página de códigos desea utilizar al transformar datos en tiempo de ejecución? Si desea utilizar una página de códigos específica para la aplicación diferente a la página de códigos para la región de CICS, utilice el parámetro **CCSID**.
 - ¿Desea soportar un subconjunto de operaciones declaradas en la descripción de servicio web? Si dispone de una descripción de servicio web muy grande, y desea que la aplicación de solicitante de servicio soporte únicamente un determinado número de operaciones, utilice el parámetro **OPERATION** para listar las que desea. Cada operación debe estar separada por un espacio y debe distinguir entre mayúsculas y minúsculas.
 - ¿Dónde está almacenado el documento WSDL? Si el documento WSDL que desea utilizar como entrada para DFHWS2LS está almacenado en un servidor WSRR, puede recuperarlo ejecutando DFHWS2LS con determinados parámetros especificados. Utilice el parámetro **WSRR-SERVER** para especificar la ubicación del servidor WSRR y utilice el parámetro **WSRR-NAME** para especificar el nombre del documento WSDL que desea recuperar. Para obtener información sobre otros parámetros en DFHWS2LS que desee utilizar para interactuar con WSRR, consulte “DFHWS2LS: Conversión de WSDL a lenguaje de alto nivel” en la página 506.
 - Si desea recuperar el documento WSDL de un servidor WSRR, ¿desea hacerlo utilizando una conexión segura? Puede utilizar el cifrado de capa de sockets seguros (SSL) con el asistente de servicios web para interoperar de forma segura con WSRR. Por ejemplo, consulte Example of how to use SSL with the web services assistant and WSRR.

No especifique parámetros como **PROGRAM** , **URI** , **TRANSACTION** y **USERID** cuando utilice DFHWS2LS. Estos parámetros se aplican sólo a una aplicación de proveedor de servicios y, si se especifica, hacen que se genere un archivo de enlace de servicio web de modalidad de proveedor. Además del archivo de enlace de servicio web, el programa genera una estructura de datos de lenguaje.
2. Compruebe el archivo de registro para ver si se producen problemas cuando DFHWS2LS genera el archivo de enlace y las estructuras de lenguaje. Puede que CICS no soporte algunos elementos u opciones en la descripción de servicio web. Si se emiten mensajes de aviso o error, lea el consejo que se proporciona y lleve a cabo la acción adecuada. Es posible que tenga que volver a ejecutar el programa por lotes.

3. Copie el archivo de enlace de servicio web en el directorio de recogida de la interconexión de modalidad de solicitante que desea utilizar para la aplicación de servicio web.
4. Asegúrese de que el recurso PIPELINE está configurado para aplicaciones de solicitante de servicio. El valor del parámetro **MODE** muestra si la interconexión soporta aplicaciones de servicio web de solicitante o proveedor.
5. Asegúrese de que se soporta el protocolo SOAP correcto en la interconexión para el servicio web. El parámetro **SOAPLEVEL** indica qué versión se soporta. En la modalidad de solicitante de servicio, el enlace del servicio web debe coincidir con la versión de SOAP que se soporta en la interconexión. No puede instalar un servicio web con un enlace SOAP 1.1 en una interconexión de solicitante de servicio que soporta SOAP 1.2.
6. Asegúrese de que el tiempo de espera configurado para la interconexión es adecuado para la aplicación de solicitante de servicio. El tiempo de espera se visualiza como el valor del atributo RESPWAIT en el recurso PIPELINE. Si no se especifica un tiempo de espera en la interconexión, se utiliza el valor predeterminado para el transporte.
 - El tiempo de espera predeterminado para HTTP es de 10 segundos.
 - El tiempo de espera para WebSphere MQ es de 60 segundos.

Cada transacción en la región de CICS tiene un tiempo de espera de asignador. Si este valor es menor del valor determinado para cualquier protocolo, el tiempo de espera se produce con el asignador.

7. Opcional: Copie la descripción de servicio web en el mismo directorio de recogida que el archivo de enlace del servicio web, para que pueda activar la validación para el servicio web en tiempo de ejecución.
8. Cree el recurso WEBSERVICE. Este recurso encapsula el archivo de enlace de servicio web en CICS y se utiliza en tiempo de ejecución.

Puede hacer esto de diferentes formas:

 - a. Utilizando el mandato **PIPELINE SCAN** para crear dinámicamente el recurso WEBSERVICE.
 - b. Definición del recurso. Si utiliza CICS Explorer para definir un recurso WEBSERVICE en un paquete CICS, puede elegir importar un archivo de enlace de servicio web y un documento WSDL o archivo de archivado WSDL e incluirlos en el paquete. También puede generar definiciones URIMAP para soportar el servicio web y empaquetarlas en un paquete. Para obtener más ayuda utilizando CICS Explorer para crear y editar recursos en paquetes CICS, consulte Working with bundles in the CICS Explorer product documentation.
9. Escriba un programa derivador que pueda sustituir para la lógica de comunicaciones. Utilice la estructura de datos de lenguaje generada en el paso 1 para escribir su programa derivador. Utilice un mandato **EXEC CICS INVOKE SERVICE** en el programa derivador para comunicarse con el servicio web. El mandato incluye estas opciones:
 - El URI del servicio web
 - La operación por la que se está invocando el servicio web

Cuando invoca el servicio web, puede especificar un recurso URIMAP que contiene la información sobre el URI del servicio web. Puede especificar esta información directamente en el mandato INVOKE SERVICE en lugar de utilizar un recurso URIMAP. Sin embargo, utilizar un recurso URIMAP significa que no necesita recompilar las aplicaciones si el URI de un proveedor de servicios cambia. Con un recurso URIMAP también puede elegir implementar la técnica de agrupación de conexiones, donde CICS mantiene abierta la conexión de

cliente después de utilizarla, por lo tanto, la aplicación puede volver a utilizarla para solicitudes posteriores, o puede volver a utilizarla otra aplicación que invoque el mismo servicio. El mandato PIPELINE SCAN no crea recursos URIMAP para un solicitante de servicio, así que debe definir el recurso URIMAP siguiendo las instrucciones en Creating a URIMAP resource for CICS as a HTTP client.

Resultados

Cuando haya creado correctamente los recursos CICS, la creación de la aplicación de solicitante de servicio está completa.

Creación de un servicio web utilizando un conjunto de herramientas

En lugar de utilizar el JCL del asistente de servicios web, puede utilizar IBM Developer for z Systems o escribir su propio programa Java para crear los archivos necesarios en CICS.

Procedimiento

1. Tiene dos opciones:
 - Utilizar la herramienta IBM Developer for z Systems para crear un archivo de enlace de servicio web y estructuras de lenguaje o descripción de servicio web. Para obtener más información acerca de esta herramienta, consulte Rational Developer for System z.
 - Escriba su propio programa Java, utilizando la API proporcionada, para invocar el asistente de servicios web. Esta API se describe en el Javadoc Asistente de servicio web: Referencia de clase. Incluye comentarios que explican las clases, y se proporciona el código de ejemplo para mostrar cómo puede llamar al asistente de servicios web. El Javadoc también contiene una lista completa de los archivos JAR necesarios y su ubicación en z/OS UNIX. Puede ejecutar el programa Java en la plataforma z/OS , Windows , o Linux. Si ejecuta el programa en Windows o Linux , transfiera el archivo de enlace de servicios web generado a un directorio de recogida adecuado en modalidad binaria utilizando FTP o un proceso equivalente.
2. Opcional: Si está generando una descripción de servicio web desde una estructura de lenguaje, revise el archivo y realice cualquier personalización necesaria. Para obtener más información, consulte el apartado “Personalización de los documentos de descripción de servicios web generados” en la página 584.
3. Despliegue el archivo de enlace de servicio web generado en un directorio de recogida de interconexión adecuado.
4. Opcional: Copie la descripción de servicio web en un directorio de recogida de la interconexión, para que pueda llevar a cabo la validación del servicio web para comprobar que todo está funcionando como esperaba.
5. Si ha comenzado con la descripción de servicio web, escriba un proveedor de servicios o un programa de aplicación de solicitante para interactuar con las estructuras de lenguaje generadas.
6. Ejecute un mandato **PIPELINE SCAN** para crear automáticamente los recursos CICS necesarios.

Creación de sus propias aplicaciones de servicio web preparadas para XML

Si decide no utilizar las correlaciones de datos proporcionados por CICS, puede escribir sus propias aplicaciones de datos preparadas para XML de dos formas. Puede utilizar el parámetro **XML-ONLY** en DFHWS2LS o puede escribir su propia aplicación sin utilizar ninguna de las herramientas. Utilizar el parámetro **XML-ONLY** es la forma más sencilla de configurar el proceso de interconexión de CICS para pasar datos XML a la aplicación que se va a manejar.

Acerca de esta tarea

Escribir sus propias aplicaciones preparadas para XML implica la escritura de un código para analizar y generar documentos XML. Una forma de escribir su propia aplicación preparada para XML utiliza las sentencias XML PARSE y XML GENERATE en COBOL. Otra forma de escribir sus propias aplicaciones preparadas para XML utiliza otras herramientas de IBM; por ejemplo, puede utilizar IBM Developer for z Systems para que la herramienta genere programas convertidores COBOL XML que se pueden invocar desde sus aplicaciones.

Creación de una aplicación de proveedor de servicios preparada para XML

La aplicación de proveedor de servicios preparada para XML debe funcionar con los contenedores que le pasan y manejar la conversión de datos entre el XML y el lenguaje de programa.

Acerca de esta tarea

Los pasos siguientes le guían a través de la creación de la aplicación preparada para XML, incluida la decisión sobre el uso de cualquier conjunto de herramientas CICS.

Procedimiento

1. Decida si desea generar un archivo de enlace de servicio web para la aplicación preparada para XML utilizando DFHWS2LS. La ventaja de generar un archivo de enlace de servicio web es que puede utilizar servicios CICS, como la validación, para probar el servicio web y la supervisión de CICS utilizando salidas de usuario globales.
 - Si desea generar un archivo de enlace de servicio web, ejecute DFHWS2LS especificando el parámetro **XML-ONLY** y un **MINIMUM-RUNTIME-LEVEL** de 2.1 o superior. El archivo de enlace de servicio web habilita el programa de aplicación para trabajar directamente con el contenido del contenedor DFHWS-BODY. En todos los demás aspectos, el archivo de enlace generado comparte las mismas características de despliegue y el mismo comportamiento en tiempo de ejecución que un archivo generado sin el parámetro **XML-ONLY**, incluido el análisis del XML durante el enlace del mensaje SOAP. Para impedir este análisis, no debe especificar “Manejadores de mensajes SOAP” en la página 145 en el archivo de configuración de interconexión.
 - Si no desea utilizar un archivo de enlace de servicio web, configure la interconexión de proveedor de servicios para que la solicitud de servicio web alcance la aplicación preparada para XML. Puede configurar el controlador de terminal en el archivo de configuración de interconexión para utilizar el

programa de aplicación preparado para XML o crear un manejador de mensajes que conmuta dinámicamente a la aplicación dependiendo del URI que se recibe en la interconexión.

2. Escriba la aplicación para manejar la solicitud de servicio que que hay en los siguientes contenedores:

DFHWS-BODY

El contenido del cuerpo SOAP para una solicitud SOAP cuando la interconexión incluye un manejador de mensajes SOAP proporcionado por CICS.

DFHREQUEST

La solicitud completa, incluido el sobre para una solicitud SOAP, recibida de la interconexión.

DFHWS-XMLNS

Una lista de pares nombre-valor que correlaciona prefijos de espacio de nombres con espacios de nombres para el contenido XML de la solicitud.

DFHWS-SOAPACTION

La cabecera SOAPAction asociada al mensaje SOAP en el contenedor DFHWS-BODY.

Cuando codifica mandatos de API para trabajar con los contenedores, no especifique la opción CHANNEL, porque todos los contenedores están asociados al canal actual (el canal que se ha pasado al programa). Si necesita saber el nombre del canal, utilice el mandato **EXEC CICS ASSIGN CHANNEL**.

3. Opcional: La aplicación también utiliza contenedores adicionales disponibles para los manejadores de mensajes en la interconexión, así como cualquier otro contenedor que creen los manejadores de mensajes como parte de su proceso. Para obtener una lista completa de los contenedores, consulte “Contenedores utilizados en la interconexión” en la página 150.
4. Cuando la aplicación ha procesado la solicitud, construya una respuesta de servicio web utilizando los siguientes contenedores:

DFHRESPONSE

El mensaje de respuesta completo que se va a pasar a la interconexión. Utilice este contenedor si no utiliza SOAP para los mensajes, o si desea compilar el mensaje SOAP completo, incluido el sobre, en su programa en lugar de utilizar el manejador de mensajes SOAP proporcionado por CICS.

Si proporciona un cuerpo SOAP en el contenedor DFHWS-BODY, DFHRESPONSE se ignora.

DFHWS-BODY

Para una respuesta SOAP de salida, el contenido del cuerpo SOAP. Proporcione este contenedor cuando el controlador de terminal de la interconexión es un manejador de mensajes SOAP proporcionado por CICS. El manejador de mensajes construye el mensaje SOAP completo conteniendo el cuerpo.

El programa debe crear este contenedor, aunque la solicitud y respuesta sean idénticas. Si no lo hace, CICS emite un error de servidor interno.

También puede utilizar cualquier otro contenedor para pasar la información que necesita la interconexión para procesar la respuesta de salida.

Si el servicio web no devuelve una respuesta, debe devolver el contenedor DFHNORESPONSE para indicar que no hay respuesta. El contenido del contenedor no es importante, porque el manejador de mensajes sólo comprueba si el contenedor está presente o no.

5. Cree un recurso URIMAP. Si está utilizando el parámetro **XML-ONLY** y ha especificado un valor para el parámetro **URI** de DFHWS2LS, el URIMAP se crea automáticamente durante el proceso PIPELINE SCAN.

Creación de una aplicación de solicitante de servicio preparada para XML

La aplicación de solicitante de servicio web preparada para XML maneja la conversión de datos entre el XML y el lenguaje de programación y llena los contenedores de control de la interconexión.

Antes de empezar

Puede escribir su propia aplicación de solicitante de servicio preparada para XML utilizando el parámetro **XML-ONLY** en DFHWS2LS o puede escribirla sin utilizar ninguna de las herramientas. La forma más sencilla de escribir su propia aplicación de solicitante de servicio preparada para XML es utilizando el parámetro **XML-ONLY** en DFHWS2LS; el parámetro **XML-ONLY** está disponible en el nivel de tiempo de ejecución 2.1 y posterior.

Acerca de esta tarea

Utilizar el parámetro **XML-ONLY** da como resultado la generación de un archivo WSBIND que indica a CICS que la aplicación trabajará directamente con el contenido del contenedor DFHWS-BODY. El archivo WSBIND generado debe instalarse en una PIPELINE modalidad de solicitante para crear un recurso WEBSERVICE modalidad de solicitante. La aplicación debe generar XML para el cuerpo de la solicitud de servicio web y almacenarlo en el contenedor DFHWS-BODY. Debe invocar el mandato **EXEC CICS INVOKE SERVICE**. El mensaje de salida se envía al proveedor de servicios web. El cuerpo del mensaje de respuesta también está en el contenedor DFHWS-BODY una vez se completa la llamada.

El XML de los mensajes de respuesta se analiza durante el manejo de mensajes SOAP. Para impedir este análisis, no debe especificar SOAP message handlers en el archivo de configuración de interconexión.

Las aplicaciones de solicitante preparadas para XML pueden recibir mensajes de error SOAP devueltos de la aplicación de modalidad de proveedor remota. En este caso, la aplicación de solicitante es responsable de interpretar el error SOAP y de distinguirlo de un mensaje de respuesta normal. Si se utiliza el mandato **INVOKE SERVICE** con un **XML-ONLY** WEBSERVICE, CICS no establece el código de respuesta que se utiliza normalmente para indicar que se ha recibido un error SOAP.

Si está escribiendo la aplicación de solicitante de servicio preparada para XML sin utilizar la opción **XML-ONLY**, complete los pasos siguientes:

Procedimiento

1. Cree un canal y llénelo con contenedores. Los contenedores de control deben llenarse en modalidad CHAR. Proporcione la siguiente información en cada contenedor:

DFHWS-PIPELINE

El nombre del recurso PIPELINE utilizado para la solicitud de salida.

DFHWS-URI

El URI del servicio web de destino

DFHWS-BODY

Para una solicitud SOAP de salida, el contenido del cuerpo SOAP. Proporcione este contenedor cuando la interconexión incluye un manejador de mensajes SOAP proporcionado por CICS. El manejador de mensajes construye el mensaje SOAP completo que contiene el cuerpo.

DFHREQUEST

El mensaje de solicitud completo que pasa a la interconexión. Utilice este contenedor si no utiliza SOAP para los mensajes o si desea compilar el mensaje SOAP completo, incluido el sobre, en su programa. La interconexión no debe incluir un manejador de mensajes SOAP proporcionado por CICS para evitar que se envíen manejadores SOAP duplicados en el mensaje de salida.

Si proporciona un cuerpo SOAP en el contenedor DFHWS-BODY, DFHREQUEST debe estar vacío. Si proporciona contenido en DFHWS-BODY y DFHREQUEST, CICS utiliza DFHREQUEST.

DFHWS-XMLNS

Una lista de pares nombre-valor que correlaciona prefijos de espacio de nombres con espacios de nombres para el contenido XML de la solicitud.

DFHWS-SOAPACTION

La cabecera SOAPAction que se va a añadir al mensaje SOAP especificado en el contenedor DFHWS-BODY.

Consejo: Si añade el contenedor DFHWS-NOABEND al canal, cuando DFHPIRT esté invocando cualquier terminación anómala no se emitirá desde dentro de DFHPIRT. Esto es útil si está ejecutando un programa C/C++ porque puede manejar errores a través del contenedor DFHERROR.

2. Enlace con un programa DFHPIRT. Utilice este mandato:

```
EXEC CICS LINK PROGRAM(DFHPIRT) CHANNEL(
  userchannel
)
```

donde *userchannel* es el canal que contiene los contenedores. El mensaje de salida se procesa por los manejadores de mensajes y los programas de proceso de cabeceras en la interconexión y se envía al proveedor de servicios.

3. Recupere los contenedores que contienen la respuesta de servicio web desde el mismo canal. La respuesta del proveedor de servicio web puede ser una respuesta satisfactoria o un error SOAP. La aplicación de solicitante de servicio web debe ser capaz de manejar ambos tipos de respuesta desde el proveedor de servicios. La respuesta completa está en los siguientes contenedores:

DFHRESPONSE

La respuesta completa, incluido el sobre para una respuesta SOAP, recibida desde el proveedor de servicios web.

DFHWS-BODY

Cuando la interconexión incluye un manejador de mensajes SOAP proporcionado por CICS, el contenido del cuerpo SOAP.

DFHERROR

Información de error de la interconexión.

Nota: En algunos casos de error, es posible que DFHWS-BODY no se actualice. Debe comprobar DFHRESPONSE para un error SOAP.

Uso de Java con servicios web

Puede utilizar Java para crear aplicaciones de servicios web. Para crear estas aplicaciones se utilizan técnicas diferentes a las utilizadas con otros lenguajes de programación.

Para la mayoría de los lenguajes de programación que no son Java, utilice los asistentes de servicios web para habilitar aplicaciones. Utilizar el asistente de servicios web significa que CICS convertirá los datos del servicio web en un formato adecuado para la aplicación y los colocará en un contenedor o COMMAREA. Puede utilizar el asistente de servicios web con aplicaciones Java; sin embargo, las siguientes tareas proporcionan más métodos adecuados para la creación de servicios web de Java para aplicaciones Java.

Despliegue de un servicio web de modalidad de proveedor Java en un servidor JVM de Axis2

Puede desplegar una aplicación Axis2 como servicio web de modalidad de proveedor en CICS. Estas aplicaciones se generan normalmente utilizando JAX-WS y se pueden alojar en una interconexión habilitada para Java.

Es posible que desee desplegar aplicaciones Java utilizando este método por una de las siguientes razones:

- Tiene una inversión utilizando las interfaces del manejador de Axis2.
- Puede utilizar una configuración de interconexión de CICS.

Nota: Las aplicaciones de estilo Axis2 no utilizan recursos WEBSERVICE. Interactúan con CICS utilizando el modelo de programación Axis2 y, por lo tanto, no pueden utilizar parte del soporte de servicios web de CICS. Los siguientes servicios no están soportados completamente para aplicaciones estilo Axis2:

- SOAPFAULT CREATE
- WSACONTEXT GET
- “Contenedor DFHWS-OPERATION” en la página 165
- “Contenedor DFHWS-MEP” en la página 164
- “Contenedor DFHWS-USERID” en la página 170
- “Contenedor DFHWS-TRANID” en la página 166
- Seguridad de los servicios web

Antes de empezar

Debe tener una aplicación Java adecuada para el despliegue en Axis2, por ejemplo, una aplicación POJO que utiliza JAX-WS. Para esta tarea, se utiliza la siguiente aplicación POJO como ejemplo:

```

/**
 * Simple example
 */
@javax.jws.WebService(targetNamespace = "com.ibm.cics.example", name = "pojoExample")
public class TestAxis2
{
    public String getMessage(String input)
    {
        return "CICS got this: '" + input + "'";
    }
}

```

Esta aplicación especifica el espacio de nombres XML que se utiliza para generar el WSDL, y un nombre para asociarlo al servicio web.

Debe compilarse el código Java para esta aplicación, y ejecutarse el generador JAX-WS, para empaquetar la aplicación en un archivo jar denominado TestAxis2.jar . Puede hacerlo emitiendo el código siguiente:

```

javac TestAxis2.java
wsngen -cp . TestAxis2 -wsdl
jar -cvf TestAxis2.jar *

```

El generador JAX-WS también crea un documento WSDL y los enlaces utilizados por Axis2.

Acerca de esta tarea

Para desplegar un servicio web Axis2, debe crear la infraestructura de interconexión para los servicios web. Cuando ha creado la interconexión, puede crear los servicios web. Puede reutilizar la interconexión creada para tantos servicios web como necesite. Los pasos siguientes describen cómo crear la interconexión y los servicios web.

Nota: No se crea ni se instala un recurso WEBSERVICE como parte de esta tarea.

Procedimiento

1. Cree la infraestructura de interconexión.
 - a. Cree una infraestructura de servicio para una interconexión Java. Para obtener más información, consulte el apartado “Creación de la infraestructura de CICS para un proveedor de servicios SOAP” en la página 64.
 - b. Cree un repositorio Axis2. Para hacer esto, cree una copia del repositorio proporcionado ubicado en \$CICS_HOME/lib/pipeline/repository.
 - c. Añada el elemento <repository> al archivo de configuración de interconexión. Este elemento debe especificar el nombre del repositorio Axis2 que ha creado.
 - d. Cree y habilite un recurso PIPELINE.
2. Para cada servicio web asociado con la interconexión, repita los pasos siguientes para crear el servicio web.
 - a. Despliegue la aplicación Axis2 en el repositorio Axis2. Por ejemplo, el archivo jar creado en el ejemplo debe desplegarse en un directorio denominado servicejars en el directorio de repositorio. Debe crear este directorio si no existe.
 - b. Defina e instale un recurso URIMAP para el servicio web. El recurso URIMAP debe especificar el URI y el recurso PIPELINE asociado al servicio web. El URI debe seguir las convenciones de denominación de Axis2 para

los URI. La convención de denominación de Axis2 predeterminada es: / *name_of_service* Service. *name_of_port* Port/ *suffix* , donde *name_of_service* es el nombre del servicio web en el WSDL, *name_of_port* es el nombre del puerto en el WSDL y *suffix* es un sufijo opcional que puede definir. En el ejemplo anterior, se puede utilizar el recurso URIMAP siguiente:

```
Urimap : EXAMPLE
Group : EXAMPLE
SStatus : Enabled
USAge : Pipeline
SCHEME : HTTP
Port : No
HOST : *
Path : /TestAxis2Service.pojoExamplePort/example/TestAxis2
Transaction : CPIH
PIpeline : EXAMPLE
```

Este ejemplo da por supuesto que el recurso PIPELINE utilizado se denomina EXAMPLE.

Qué hacer a continuación

Pruebe que los servicios web se ejecutan correctamente.

Creación de un servicio web Java que genera y analiza XML

Puede crear aplicaciones Java que analizan y generan XML ellas mismas. Estas aplicaciones son coherentes con aplicaciones preparadas para XML escritas en otros lenguajes de programación, pero se benefician de utilizar tecnologías Java estándar para procesar el XML.

Procedimiento

1. Cree un recurso XML-ONLY WEBSERVICE. Para obtener más información, consulte “Creación de una aplicación de solicitante de servicio preparada para XML” en la página 593 o “Creación de una aplicación de proveedor de servicios preparada para XML” en la página 591.
2. Escriba un servicio web Java que pueda analizar y generar XML para el cuerpo del mensaje SOAP. Puede utilizar varias herramientas, como la biblioteca Java 6 Java Architecture for XML Binding (JAXB), para que le ayude a crear un servicio web Java con estas prestaciones.
3. Opcional: Si está utilizando una interconexión de proveedor y desea añadir la prestación para que se devuelva un mensaje de error SOAP al solicitante, utilice la clase JCICS SoapFault para emitir el mandato **EXEC CICS SOAPFAULT CREATE**.
4. Opcional: Si está utilizando una interconexión de solicitante, utilice una clase JCICS Service para emitir el mandato **EXEC CICS INVOKE SERVICE**.

Creación de un servicio web Java que tiene una interfaz COBOL

Puede crear aplicaciones Java que interactúan con CICS utilizando las mismas técnicas utilizadas en otros lenguajes de programación. Para crear estas aplicaciones, debe escribir o generar código Java para crear datos de estilo de contenedor o COMMERA estructurados.

Procedimiento

1. Utilice DFHWS2LS para crear estructuras de lenguaje COBOL para el servicio web.

2. Escriba un servicio web Java que genere y analice estructuras de lenguaje COBOL. Para obtener más información sobre las herramientas que permiten a los programas Java acceder a los datos de aplicación CICS existentes y enlaces a ejemplos de cómo crear un servicio web Java que pueda generar y analizar estructuras de lenguaje COBOL, consulte *Interacting with structured data from Java*.
3. Opcional: Si está utilizando una interconexión de proveedor y desea añadir la prestación para que se devuelva un mensaje de error SOAP al solicitante, utilice la clase JCICS SoapFault para emitir el mandato **EXEC CICS SOAPFAULT CREATE**.
4. Opcional: Si está utilizando una interconexión de solicitante, utilice la clase de servicio JCICS para funcionar en conjunto con CICS SERVICE API y emitir el mandato **EXEC CICS INVOKE SERVICE**.

Despliegue de un servicio web JAX-WS modalidad solicitante

Puede desplegar una aplicación JAX-WS como servicio web modalidad solicitante en CICS. Sin embargo, estas aplicaciones no utilizan el mandato **EXEC CICS INVOKE**, en lugar de eso interactúan con los servicios web remotos utilizando JAX-WS.

Antes de empezar

Debe tener un servidor JVM configurado para soportar OSGi. Para obtener más información, consulte *Setting up a JVM server*.

Acerca de esta tarea

La ventaja de desplegar una aplicación JAX-WS como servicio web modalidad solicitante es que crea una aplicación de solicitante de servicio web independiente de plataforma, que utiliza zEnterprise Application Assist Processor (zAAP). Utilizando zAAP puede reducir el coste de transacciones; para obtener más información, consulte la publicación de IBM Redbooks: *zSeries Application Assist Processor (zAAP) Implementation*.

Procedimiento

1. Cree una aplicación de solicitante de servicio web en Java y utilice una API adecuada, como la API de Java para servicios web XML (JAX-WS), para llamar al servicio web remoto.
2. Opcional: Si utiliza JAX-WS para iniciar un servicio web remoto, también debe utilizar JAX-WS para generar mensajes SOAP, gestionar la comunicación de red y procesar la respuesta SOAP.
3. Despliegue la aplicación Java e instálela en el servidor JVM.

Qué hacer a continuación

Pruebe que los servicios web se inician correctamente.

Despliegue de un servicio web de modalidad de proveedor Java en un servidor Liberty JVM

Puede desplegar una aplicación web como un servicio web de modalidad de proveedor en un servidor Liberty JVM. Estas aplicaciones se crean utilizando los estándares de Java JAX-WS y JAXB. Este tema se aplica solo a Liberty en modalidad integrada de CICS.

Acerca de esta tarea

CICS TS V5.3 incluye el WebSphere Application Server Liberty Profile (WLP) más reciente que proporciona características para la API de Java para servicios web XML (JAX-WS) y la arquitectura de Java para el enlace XML (JAXB). Juntas estas tecnologías le permiten escribir servicios web SOAP en Java como parte de una aplicación CICS. El siguiente artículo mostrará cómo configurar Eclipse, probar un proyecto de servicios web de ejemplo, desplegar el ejemplo en CICS, modificar el ejemplo para utilizar JCICS y probarlo con el explorador de servicios web. Para obtener más información, consulte el artículo de CICS DevCenter, Ejemplo de servicio web JAX-WS para Liberty.

Es posible que desee desplegar aplicaciones Java utilizando este método por una de las siguientes razones:

- Desea crear servicios web en Java.
- Tiene documentos WSDL complicados que será difícil gestionar utilizando asistentes de servicios web de CICS.
- Desea descargar el manejo de la aplicación de servicio web para zEnterprise Application Assist Processor (zAAP).

Nota: Las aplicaciones web desplegadas en un servidor Liberty JVM no utilizan los recursos WEBSERVICE o TCPIPService. Interactúan con solicitudes web utilizando el proceso de escucha Liberty HTTP y, por lo tanto, no se pueden utilizar los servicios del soporte de servicios web CICS.

Validación de mensajes SOAP

Cuando utiliza los servicios web de CICS, puede especificar que los mensajes SOAP se van a validar para asegurarse de que se ajustan al esquema contenido en la descripción del servicio web. Puede validar las aplicaciones en modalidad tanto de proveedor como de solicitante.

Antes de empezar

Durante el desarrollo y prueba del despliegue de servicio web, una validación completa ayuda en la detección de problemas en el intercambio de mensajes entre un solicitante de servicio y un proveedor de servicios. Sin embargo, una validación completa de los mensajes SOAP tiene una sobrecarga considerable y no es aconsejable validar los mensajes en una aplicación de producción completamente probada.

CICS utiliza un programa Java para validar mensajes SOAP. Por lo tanto, debe tener el soporte de Java habilitado en la región de CICS para validar mensajes SOAP.

Acerca de esta tarea

El mensaje SOAP se valida antes de que se transforme en una estructura de datos de aplicación y cuando se genera un mensaje SOAP desde la estructura de datos de la aplicación. El mensaje SOAP se valida utilizando el esquema XML en WSDL y se valida de nuevo en los requisitos de transformación de CICS. Puede utilizar el archivo WSDL especificado en el atributo **WSDLFILE** del recurso WEBSERVICE o un archivo WSDL contenido en el archivo .zip especificado en el atributo **ARCHIVEFILE** del recurso WEBSERVICE. Si se especifican ambos atributos, se utiliza el archivo WSDL en el archivo de archivado especificado en el atributo **ARCHIVEFILE**.

Cuando la validación se desactiva, CICS no utiliza el programa Java. CICS valida los mensajes SOAP sólo en la medida que sean necesarios para confirmar que contienen XML bien formado y para transformarlos. Por lo tanto, un mensaje SOAP puede validarse correctamente pero puede fallar en el entorno de tiempo de ejecución y viceversa.

Procedimiento

1. Configure un servidor OSGi JVM en la región de CICS. La validación SOAP que utiliza DFHPIVAL sólo se ejecutará en un marco de trabajo OSGi, no en un JVM de perfil de Axis2 o Liberty.
 - a. Instale el DFH\$JVMS del servidor JVM de ejemplo en el grupo DFH\$OSGI o cree su propio servidor JVM. Para obtener más información, consulte el apartado Setting up a JVM server.
 - b. Si ha creado su propio servidor JVM, modifique la definición de programa DFHPIVAL en el grupo DFHPIVAL para hacer referencia al nombre del recurso JVMSERVER. La definición DFHPIVAL no está bloqueada y se puede editar. De forma predeterminada, la definición hace referencia a DFH\$JVMS.
2. Asegúrese de que tiene una descripción de servicio web asociada al recurso WEBSERVICE. Esta asociación se crea para los recursos WEBSERVICE que se crean automáticamente cuando un archivo WSDL o un archivo .zip que contiene uno o más archivos WSDL está presente en el directorio de recogida de la interconexión durante una exploración de interconexión.

Para las definiciones WEBSERVICE que se crean con RDO, se especifica la descripción de servicio web con el atributo WSDLFILE.
3. Active la validación de servicio web especificando el atributo **VALIDATION=YES** del recurso WEBSERVICE. Puede especificar si la validación es necesaria cuando define el recurso, y puede cambiar este valor después de que se instala el recurso.

Resultados

Compruebe el registro del sistema para averiguar si el mensaje SOAP es válido. El mensaje DFHPI1002 indica que el mensaje SOAP se ha validado correctamente, y el mensaje DFHPI1001 indica que la validación ha fallado.

Qué hacer a continuación

Desactive la validación cuando ya no la necesite.

Manejo de datos proporcionados por la aplicación no válidos ni inicializados

Cómo tolerar datos no válidos que se encuentran durante la transformación entre JSON y datos de aplicación.

Un servicio habilitando una aplicación existente da como resultado que CICS tenga en cuenta el contenido de COMMAREA o de los contenedores que está utilizando esa aplicación. El archivo WSBIND que se despliega en CICS contiene información sobre el formato de datos, incluidos los nombres, ubicación y tipo de datos de cada campo individual. CICS utiliza esta información para facilitar la transformación entre la representación de datos XML/JSON y los datos de aplicación.

Si los datos de aplicación no son coherentes con la información almacenada en el archivo WSBIND, CICS informa de un problema y no puede transformar los datos. Por ejemplo, se cambia una interfaz de aplicación pero el archivo WSBIND no se vuelve a generar ni a desplegar. Este tipo de problema a menudo da como resultado un mensaje de error DFHPI1010.

Se puede producir una variante más sutil de esta condición en escenarios donde la aplicación ha dejado campos que no se han inicializado deliberadamente. Por ejemplo, un campo decimal empaquetado firmado puede contener valores bajos (bytes nulos). Cuando CICS procesa ese campo, no puede determinar que el valor no válido no se ha inicializado deliberadamente. CICS muestra el mensaje de error DFHPI1010 y la transformación de datos falla. Las aplicaciones pueden dejar campos sin inicializar por muchas razones. Por ejemplo,

- Una estructura de lenguaje puede describir los formatos de datos de entrada y salida, pero el campo no inicializado sólo era para la entrada.
- La aplicación puede detectar un error, establecer un código de respuesta y volver sin los demás campos de salida inicializados.
- La lógica condicional dentro de la aplicación puede controlar si el campo se considera importante.

La respuesta ideal a este escenario es que se cambie la aplicación para garantizar que no se pasan valores no válidos a CICS para procesarlos. Una opción alternativa es establecer la opción DATA-SCREENING del asistente en DISABLED al mismo tiempo que se genera el archivo WSBIND. Esta opción hace que CICS tolere los valores erróneos proporcionados por la aplicación y sustituya esos valores por un valor predeterminado, normalmente cero. Estas opciones pueden hacer que la habilitación de servicio de aplicaciones existentes sea más simple, pero hace que la detección de errores sea más difícil, y debe utilizarse con cuidado. Si el cribado de datos está inhabilitado, es más probable que CICS no haya detectado los errores de datos generados por la aplicación.

Se puede experimentar una condición relacionada donde las matrices generadas por la aplicación quedan parcialmente sin llenarse. Por ejemplo, tenga en cuenta una matriz de 1.000 registros de datos; la aplicación puede inicializar los primeros 10 registros. CICS genera una representación JSON o XML de datos y llena todos los campos, aunque 990 estén vacíos. La respuesta ideal para este problema es introducir una cláusula OCCURS DEPENDING ON en la aplicación para indicar de forma precisa cuántos registros se tienen que llenar. Una alternativa es utilizar la opción TRUNCATE=NULL-ARRAYS de los asistentes cuando se genera el archivo WSBIND. Esta opción indica a CICS que intente detectar los datos no inicializados y truncar la matriz en ese momento. El uso de esta opción puede dar como resultado datos JSON/XML elegantes que se generan a partir de datos de aplicación ambiguos y no inicializados, pero introduce el riesgo de pérdida de datos accidental si el contenido de la matriz no se distingue de los datos no inicializados.

La mejor solución es tener aplicaciones que producen datos no ambiguos que son coherentes completamente con la estructura de lenguaje que describe los datos. El uso de las opciones TRUNCATE=NULL-ARRAYS=ENABLED y DATA-SCREENING=DISABLED puede provocar que CICS tolere datos imperfectos proporcionados por la aplicación, pero introduce un elemento de riesgo e incertidumbre en el proceso.

El ejemplo siguiente muestra cómo utilizar estas opciones.

Ejemplo 1: tolerancia de campos decimales

Prueba de la corrección automática de datos no válidos en un campo decimal.

Acerca de esta tarea

Este escenario muestra cómo el valor predeterminado 0 se puede establecer automáticamente en campos decimales que contienen datos no válidos.

Procedimiento

1. Genere el esquema JSON y los artefactos necesarios para que CICS transforme entre datos de aplicación JSON y COBOL.

- a. Prepare el libro de copias.

```
03 BAD-DATA.  
05 NORMAL-NUM PIC 9(2).  
05 NORMAL-CHAR PIC X(3).  
05 PZONED-DECIMAL PIC S9(4) DISPLAY.  
05 NZONED-DECIMAL PIC S9(4) DISPLAY.  
05 UZONED-DECIMAL PIC 9(4) DISPLAY.  
05 NORMAL-CHAR2 PIC X(3).  
05 PBINARY PIC S9(4) BINARY.  
05 NBINARY PIC S9(4) COMP.  
05 UBINARY PIC 9(4) COMP.  
05 NORMAL-NUM2 PIC 9(3).  
05 PPACKED-DECIMAL PIC S9(4) COMP-3.  
05 NPACKED-DECIMAL PIC S9(4) COMP-3.  
05 UPACKED-DECIMAL PIC 9(4) COMP-3.  
05 NORMAL-NUM3 PIC 9(2).  
05 NORMAL-CHAR3 PIC X(3).  
05 FLOAT-ZONED PIC S9(4)V99.  
05 FLOAT-PACKED PIC S9(4)V99 COMP-3.
```

- b. Emita el JCL. Establezca DATA-SCREENING en DISABLED para activar la función.

```
//GENJSON JOB ('accounting information',name),CLASS=M,REGION=0M,  
// MSGCLASS=A,NOTIFY=&SYSUID  
//JCLLIB JCLLIB ORDER=ZZZ.A.ZOSCONN.JCL  
//LS2JS EXEC DFHLS2JS,  
// JAVADIR='/java/java7_64/J7.0_64',  
// USSDIR='cics.ts.test/tolerate',  
// PATHPREF='',  
// TMPDIR='/tmp',  
// TMPFILE=''  
//INPUT.SYSUT1 DD *  
PDSLIB=ZZZ.A.ZOSCONN.COPYBOOK  
REQMEM=BADDATA  
RESPMEM=BADDATA  
JSON-SCHEMA-REQUEST=/u/zzzz/json/ls2js/baddata1_request.json  
JSON-SCHEMA-RESPONSE=/u/zzzz/json/ls2js/baddata1_response.json  
LANG=COBOL  
LOGFILE=/u/zzzz/json/ls2js/baddata1.log  
MAPPING-LEVEL=4.1  
DATA-SCREENING=DISABLED  
CHAR-VARYING=COLLAPSE  
PGMNAME=baddata1  
URI=/baddata1  
PGMINT=COMMAREA  
WSBIND=/u/zzzz/json/ls2js/baddata1.wsbind  
/*
```

- c. Compruebe que los esquemas JSON generados (solicitud y respuesta) y los archivos WSBind se crearon correctamente.
2. Cree el programa de aplicación.
 - a. En un programa COBOL, asigne el valor a todos los campos excepto los decimales.

```

MOVE LOW-VALUE TO BAD-DATA.
MOVE 11 TO NORMAL-NUM.
MOVE 'AAA' TO NORMAL-CHAR.
MOVE 'BBB' TO NORMAL-CHAR2.
MOVE 222 TO NORMAL-NUM2.
MOVE 'CCC' TO NORMAL-CHAR3.

```

3. Defina recursos CICS.
 - a. Defina e instale el programa COBOL.
 - b. Defina e instale TCPIP SERVICE.
 - c. Basándose en el analizador, defina e instale PIPELINE.
4. Pruebe la aplicación.
 - a. Ejecute PIPELINE SCAN. Para el servicio web descrito en el archivo WSbind, utilice INSERVICE.
 - b. Envíe la solicitud JSON.


```

{"BADDATA10operation":{"bad_data":{"normal_num":12}}}

```
 - c. Compruebe la respuesta.

Resultados

Los valores predeterminados se asignan a campos no inicializados.

```

{
  "BADDATA10operationResponse": {
    "bad_data": {
      "normal_num": 11,
      "normal_char": "AAA",
      "pzoned_decimal": 0,
      "nzoned_decimal": 0,
      "uzoned_decimal": 0,
      "normal_char2": "BBB",
      "pbinary": 0,
      "nbinary": 0,
      "ubinary": 0,
      "normal_num2": 222,
      "ppacked_decimal": 0,
      "npacked_decimal": 0,
      "upacked_decimal": 0,
      "normal_num3": 0,
      "normal_char3": "CCC",
      "float_zoned": 0,
      "float_packed": 0
    }
  }
}

```

Capítulo 4. Soporte para proteger los servicios web

CICS Transaction Server for z/OS proporciona soporte para varias tecnologías relacionadas que puede utilizar para proteger los mensajes SOAP y JSON.

Algunas de estas tecnologías están disponibles como parte del protocolo HTTP, y se aplican de igual manera a SOAP y JSON. Algunas utilizan la especificación *Web Services Security (WSS): SOAP Message Security 1.0*, y sólo están disponibles para SOAP. Para obtener información en las opciones de seguridad TCP/IP y HTTP compartidas, consulte *Security for TCP/IP clients* y *Security for CICS web support*.

Para obtener información sobre el uso de aserciones SAML, consulte *Overview of SAML support*.

Seguridad de servicios web SOAP

Web Services Security (WSS): SOAP Message Security 1.0 describe el uso de *señales de seguridad* y *firmas digitales* para proteger y autenticar mensajes SOAP. Para obtener más información, consulte la especificación *WSS: Soap Message Security 1.0*.

Web Services Security protege la *privacidad* e *integridad* de mensajes SOAP, protegiendo los mensajes de una revelación no autorizada e impidiendo la modificación no autorizada y no detectada, respectivamente. WSS proporciona esta protección firmando digitalmente y cifrando elementos XML en el mensaje. Los elementos que pueden protegerse son el cuerpo y cualquier elemento del cuerpo o cabecera. Puede proporcionar diferentes niveles de protección a diferentes elementos en el mensaje SOAP.

La especificación *Web Services Trust Language* mejora aún más la Seguridad de servicios web proporcionando una infraestructura para la solicitud y emisión de señales de seguridad, y para la gestión de relaciones de confianza entre proveedores y solicitantes de servicio web. Esta ampliación para la autenticación de mensajes SOAP habilita los servicios web para validar e intercambiar señales de seguridad de diferentes tipos utilizando un tercero de confianza. Este tercero se denomina *Security Token Service (STS)*. Para obtener más información sobre Web Services Trust Language, consulte la especificación *WS-Trust Language*.

CICS Transaction Server for z/OS proporciona soporte para estas especificaciones utilizando un manejador de seguridad proporcionado por CICS en la interconexión:

- Para mensajes de salida, CICS proporciona soporte para la firma digital y el cifrado de todo el cuerpo SOAP. CICS también puede intercambiar una señal de nombre de usuario para una señal de seguridad de un tipo diferente con un STS.
- Para mensajes de entrada, CICS soporta mensajes en los que el cuerpo, o elementos del cuerpo y cabecera, se cifran y se firman digitalmente. CICS también puede intercambiar y validar señales de seguridad con STS.

CICS también proporciona una interfaz de cliente de confianza individual para que pueda interactuar con un STS sin utilizar el manejador de seguridad de CICS.

Nota: Web Services Security no cumple potencialmente con SP800-131A. La seguridad de servicios web se configura añadiendo un manejador en la

interconexión y CICS no tiene control sobre el proceso en un manejador escrito por el cliente. Si utiliza firmas digitales, puede especificar únicamente los algoritmos dsa-sha1 y rsa-sha1. Estos algoritmos no cumplen con SP800-131A. El algoritmo de cifrado de triple DES de dos claves, que se puede utilizar para cifrar un cuerpo SOAP, tampoco cumple con la especificación.

Requisitos previos para la Seguridad de servicios web

Para implementar la Seguridad de servicios web, debe aplicar estas actualizaciones a su región de CICS: instale el IBM XML Toolkit for z/OS v1.10, aplique APAR OA14956 y agregue 3 bibliotecas a la concatenación de DFHRPL.

Acerca de esta tarea

Complete los pasos siguientes antes de implementar la seguridad de servicios web:

Procedimiento

1. Instale el IBM XML Toolkit for z/OS v1.10 gratuito. Puede descargarlo desde el siguiente sitio: <http://www.ibm.com/servers/eserver/zseries/software/xml/>. Debe instalar la versión 1.10. Las versiones posteriores no funcionan con el soporte de Seguridad de servicios web en CICS.
2. Aplique ICSF APAR OA14956 si no está ya instalado en z/OS.
3. Añada las siguientes bibliotecas a la concatenación DFHRPL:
 - *hlq.SIXMLOD1*, donde *hlq* es el calificador de alto nivel de XML Toolkit.
 - *hlq.SCEERUN*, donde *hlq* es el calificador de alto nivel de Language Environment.
 - *hlq.SDFHWSLD*, donde *hlq* es el calificador de alto nivel de la instalación de CICS; por ejemplo CICSTS54.

Las dos primeras bibliotecas contienen los DLLs que el manejador de seguridad necesita durante el tiempo de ejecución. XML Toolkit proporciona y se encuentra en *hlq.SIXMLOD1*; el tiempo de ejecución de Language Environment proporciona C128N y se encuentra en *hlq.SCEERUN*.

La biblioteca *hlq.SDFHWSLD* habilita CICS para encontrar los módulos de seguridad de servicios web DFHWSSE1 y DFHWSXXX.

4. Puede que necesite aumentar el valor del parámetro de inicialización del sistema **EDSALIM**. Los tres DLLs que se cargan requieren aproximadamente 15 MB de almacenamiento EDSA.

Resultados

Si no tiene las bibliotecas especificadas, ve el siguiente mensaje:

```
CEE3501S El módulo  
nombre_módulo no se ha encontrado.
```

El *nombre_módulo* varía dependiendo de en qué biblioteca falta.

Planificación para proteger servicios web SOAP

Puede decidir la mejor forma de proteger los servicios web. CICS soporta varias opciones, incluido el manejador de mensajes de seguridad configurable y una interfaz de cliente de confianza individual.

Acerca de esta tarea

CICS implementa la seguridad de servicios web (WS-Security o WSS) a nivel de interconexión, en lugar de para cada servicio web. Responda a las siguientes preguntas para decidir la mejor forma de implementar la seguridad.

Procedimiento

1. ¿Es importante el rendimiento del proceso de interconexión? El uso de WSS para proteger los servicios web tiene un impacto significativo en el rendimiento.

La ventaja principal de la implementación de WSS es que, cifrando parte de un mensaje SOAP, puede enviar un mensaje a través de una cadena de nodos de intermediarios, todos los cuales pueden tener motivos legítimos para mirar la cabecera SOAP para tomar decisiones de procesamiento o direccionamiento, pero no tienen permiso para ver el contenido del mensaje. Cifrando únicamente esas secciones que deben ser confidenciales, se obtienen los siguientes beneficios:

- No incurra en la sobrecarga de cifrado y descifrado en cada nodo de la cadena de procesos intermedios.
- Puede enviar un mensaje confidencial a través de una red pública de nodos que no son de confianza, donde sólo el destinatario final de los datos puede entenderlo.

Como alternativa al uso de WSS, puede utilizar SSL para cifrar todo el flujo de datos.

2. Si desea utilizar WSS, ¿qué nivel de seguridad desea? El rango de opciones de la autenticación básica, donde la cabecera de mensaje incluye un nombre de usuario y contraseña, a través de la combinación de firmas digitales y cifrado del mensaje. Las opciones que soporta el manejador de seguridad de CICS se describen en “Opciones para proteger mensajes SOAP”.
3. ¿El manejador de seguridad proporcionado por CICS cumple los requisitos? Si desea realizar un proceso de seguridad más avanzada, debe escribir su propio manejador de seguridad personalizado. Este manejador debe realizar la autenticación necesaria de mensajes, directamente con RACF o utilizando Security Token Service, y gestionar el procesamiento de elementos cifrados y certificados digitales. Consulte “Escritura de un manejador de seguridad personalizado” en la página 623 para obtener más detalles.
4. ¿La interconexión incluye un manejador MTOM? Si va a habilitar el manejador MTOM y el manejador de seguridad en el archivo de configuración de interconexión, los mensajes relacionados o MIME Multipart se procesan en el modo de compatibilidad, porque el manejador de seguridad no puede analizar los elementos XOP en el cuerpo del mensaje. Este proceso puede tener un mayor efecto en el rendimiento del proceso de interconexión.

Opciones para proteger mensajes SOAP

CICS soporta la firma y cifrado de mensajes SOAP, por lo tanto, puede seleccionar el nivel de seguridad más adecuado para los datos que está enviando o recibiendo en el mensaje SOAP.

La firma y cifrado de mensajes SOAP no se soporta para aplicaciones Java de servicio web Axis2 en modalidad de proveedor o para servicios web de proveedor que se conectan a la interconexión utilizando Axis2 MessageContext.

Puede seleccionar una de estas opciones:

Autenticación de confianza

En interconexiones de proveedor de servicios, CICS puede aceptar una señal de nombre de usuario en la cabecera de mensaje SOAP como de confianza. Este tipo de señal de seguridad normalmente contiene un nombre de usuario y una contraseña, pero en este caso la contraseña no es necesaria. CICS confía en el nombre de usuario y lo coloca en el contenedor DFHWS-USERID, y el mensaje se procesa en la interconexión.

En las interconexiones de solicitante de servicio, CICS puede enviar una señal de nombre de usuario sin la contraseña en la cabecera del mensaje SOAP para el proveedor de servicios.

Autenticación básica

En modalidad de proveedor de servicios, CICS puede aceptar una señal de nombre de usuario en la cabecera de mensaje SOAP para la autenticación en mensajes SOAP de entrada. Este tipo de señal de seguridad contiene un nombre de usuario y una contraseña. CICS verifica la señal de nombre de usuario utilizando un gestor de seguridad externo, como RACF. Si es correcto, el nombre de usuario se coloca en el contenedor DFHWS-USERID y el mensaje SOAP se procesa en la interconexión. Si CICS no puede verificar la señal de nombre de usuario, se devuelve un mensaje de error SOAP al solicitante de servicio.

Las señales de nombre de usuario que contienen contraseñas no se soportan en la modalidad de solicitante de servicio o en los mensajes SOAP de salida.

Autenticación básica de HTTP

En la modalidad de proveedor de servicios, CICS puede aceptar información de autenticación básica a través de un protocolo HTTP. El solicitante de servicio utiliza una definición URIMAP para especificar qué credenciales (información de autenticación de usuario) puede capturar la salida de usuario global, XWBAUTH. XWBAUTH pasa esta información a CICS en la solicitud y CICS envía la información en una cabecera de autorización HTTP al proveedor de servicios.

Autenticación avanzada

En las interconexiones de solicitante y proveedor de servicios, puede verificar o intercambiar señales de seguridad con un Security Token Service (STS) para fines de autenticación. Esta autenticación permite a CICS aceptar y enviar mensajes que tienen las señales de seguridad en la cabecera de mensajes que no se soportan normalmente; por ejemplo, señales Kerberos o aserciones SAML.

Para un mensaje de entrada, puede seleccionar verificar o intercambiar una señal de seguridad. Si la solicitud es para intercambiar la señal de seguridad, CICS debe recibir una señal de nombre de usuario de vuelta desde el STS. Para un mensaje de salida, puede intercambiar una señal de nombre de usuario sólo para una señal de seguridad.

Firma con certificados X.509

En una modalidad de solicitante de servicio y proveedor de servicios, puede proporcionar un certificado un certificado X.509 en la cabecera de mensaje SOAP para firmar el cuerpo de un mensaje SOAP para autenticación. Este tipo de señal de solicitud se conoce como *señal de seguridad binaria*. Para aceptar señales de seguridad binarias de mensajes SOAP de entrada, la clave pública asociada al certificado debe importarse a un gestor de seguridad externo, como RACF, y asociarse al conjunto de claves que se especifica en el parámetro de inicialización del sistema

KEYRING. Para mensajes SOAP de salida, genera y publica la clave pública para los destinatarios deseados. El Integrated Cryptographic Service Facility (ICSF) se utiliza para generar claves públicas.

Cuando especifica la etiqueta asociada a un certificado digital X.509, no utilice los siguientes caracteres:

< > ; ! =

También puede incluir un segundo certificado X.509 en la cabecera y firmarlo utilizando el primer certificado. Con este segundo certificado, puede ejecutar el trabajo en CICS bajo el ID asociado al segundo certificado X.509. El certificado que está utilizando para firmar el mensaje SOAP debe estar asociado a un ID de usuario de confianza, y tener autoridad subrogada para certificar que el trabajo se ejecuta bajo una identidad diferente, la *identidad certificada*, sin que el ID de usuario de confianza tenga la contraseña asociada a esa identidad.

Cifrado

En la modalidad de solicitante de servicio y proveedor de servicios, puede cifrar el cuerpo de mensaje SOAP utilizando un algoritmo simétrico como Triple DES o AES. Un algoritmo simétrico es donde se utiliza la misma clave para cifrar y descifrar los datos. Esta clave se conoce como una *clave simétrica*. Se incluye en el mensaje y se cifra utilizando una combinación de clave pública del destinatario deseado y del algoritmo de cifrado de clave asimétrica RSA 1.5. Este cifrado le proporciona una seguridad aumentada, porque el algoritmo asimétrico es complejo y es difícil de descifrar la clave simétrica. Sin embargo, obtiene un mejor rendimiento porque la mayor parte del mensaje SOAP está cifrada con el algoritmo simétrico, que es más rápido de descifrar.

Para mensajes SOAP de entrada, puede cifrar un elemento en el cuerpo SOAP y, a continuación, cifrar el cuerpo SOAP como un todo. Esta clase de cifrado debe ser adecuada para un elemento que contiene datos confidenciales. Si CICS recibe un mensaje SOAP con dos niveles de cifrado, CICS descifra ambos niveles automáticamente. Esta clase de cifrado no está soportada para mensajes SOAP de salida.

CICS no soporta los mensajes SOAP de entrada que tiene un elemento cifrado en la cabecera de mensaje únicamente y no tiene elementos cifrados en el cuerpo SOAP.

Firma y cifrado

En la modalidad de solicitante de servicio y proveedor de servicio, puede elegir firmar y cifrar un mensaje SOAP. CICS siempre firma el cuerpo de mensaje SOAP primero y, a continuación, lo cifra. La ventaja de este método es que le proporciona integridad y confidencialidad de mensaje.

Propagación de identidad basada en ICRX

En modalidad de proveedor de servicios, puede utilizar una señal de identidad ICRX (Extended Identity Context Reference) no autenticada en las mismas circunstancias que utilizaría una señal ID de usuario de WS-Security no autenticada. Una señal de identidad ICRX es un identificador z/OS que se correlaciona con un ID de usuario. CICS resuelve la señal de identidad ICRX para un ID de usuario y coloca una copia en el contenedor DFHWS-ICRX. CICS también llena el contenedor DFHWS-USERID. Para obtener más información sobre la señal de identidad ICRX, consulte Identity propagation and distributed security.

Autenticación mediante un servicio de señales de seguridad

CICS puede interoperar con un Security Token Service (STS), como Tivoli Federated Identity Manager, para proporcionar autenticación más avanzada de servicios web.

Un STS es un servicio web que actúa como un tercero de confianza para relaciones de confianza de intermediario entre un solicitante de servicio web y un proveedor de servicios web. De forma similar a una entidad emisora de certificados en un reconocimiento SSL, el STS garantiza que el solicitante y el proveedor pueden "confiar" en las credenciales que se proporcionan en el mensaje. Esta fiabilidad se representa mediante el intercambio de señales de seguridad. Un STS puede emitir, intercambiar y validar estas señales de seguridad, y establecer relaciones de confianza, permitiendo a los servicios web de diferentes dominios de confianza comunicarse correctamente. Para obtener más detalles, consulte la especificación Web Services Trust Language.

CICS actúa como un cliente de confianza y puede enviar dos tipos de solicitud de servicio web a un STS. El primer tipo de solicitud es para validar la señal de seguridad en la cabecera del mensaje WS-Security; el segundo tipo de solicitud es para intercambiar la señal de seguridad para un tipo distinto. Estas solicitudes permiten a CICS enviar y recibir mensajes que contienen diferentes señales de seguridad desde una gran variedad de dominios de confianza, como aserciones SAML y señales Kerberos.

También puede configurar el manejador de seguridad CICS para definir cómo interactúa CICS con un STS o escribir su propio manejador de mensajes para utilizar una interfaz de cliente de confianza proporcionada por separado. Cualquiera que sea el método que seleccione, utilice SSL para garantizar la conexión entre CICS y el STS.

Cómo llama el manejador de seguridad a STS

El manejador de seguridad CICS utiliza la información en el archivo de configuración de interconexión para enviar una solicitud de servicio web a Security Token Service (STS). El tipo de solicitud que se envía depende de la acción que desee que realice el STS.

En una interconexión de proveedor de servicios

En una interconexión de proveedor de servicios, el manejador de seguridad soporta dos tipos de acciones, dependiendo de la forma que la que configure el manejador de seguridad:

- Enviar una solicitud al STS para validar la primera instancia de una señal de seguridad, o la primera señal de seguridad de un tipo especificado, en la cabecera WS-Security del mensaje de entrada.
- Enviar una solicitud al STS para intercambiar la primera instancia de una señal de seguridad, o la primera señal de seguridad de un tipo específico, en la cabecera WS-Security del mensaje de entrada, por una señal de seguridad que CICS que CICS pueda entender.

El manejador de seguridad crea dinámicamente una interconexión para enviar la solicitud de servicio web a STS. Esta interconexión existe hasta que se recibe una respuesta del STS, después de la cual se suprime. Si la solicitud es correcta, el STS devuelve una señal de identidad o el estado de validez de la señal. El manejador de seguridad coloca el ID de RACF derivado de la señal en el contenedor DFHWS-USERID.

Si el STS encuentra un error, devuelve un error SOAP al manejador de seguridad. El manejador de seguridad pasa un error de vuelta al solicitante de servicio web.

En una interconexión de solicitante de servicio

En una interconexión de solicitante de servicio, el manejador de seguridad puede solicitar únicamente intercambiar una señal con STS. El archivo de configuración de interconexión define qué tipo de señal emite el STS al manejador de seguridad.

Si la solicitud es correcta, el ID de RACF se coloca en el contenedor DFHWS-USERID y la señal se incluye en la cabecera de mensaje de salida. Si el STS encuentra un error, devuelve un error SOAP al manejador de seguridad. El manejador de seguridad pasa el error de vuelta a través de la interconexión al solicitante de servicio web.

El manejador de seguridad puede solicitar sólo un tipo de acción desde el STS para la interconexión. También puede intercambiar sólo un tipo de señal para un mensaje de solicitud de salida y se limita a gestionar la primera señal en la cabecera de mensaje WS-Security, ya sea la primera instancia o la primera instancia de un tipo específico. Estas opciones se ocupan de la mayoría de los casos de ejemplo más habituales para la utilización de un STS, pero es posible que no le proporcionen el proceso que usted necesita para manejar los mensajes de entrada y de salida.

Si desea proporcionar un proceso más específico para gestionar muchas señales en las cabeceras de mensaje de entrada o intercambiar varios tipos de señales para mensajes de salida, utilice la interfaz de cliente de confianza. Utilizando esta interfaz, puede crear un manejador de mensajes personalizado para enviar su solicitud de servicio web al STS.

Interfaz de cliente de confianza

La interfaz de cliente de confianza le permite interactuar directamente con un servicio de señales de seguridad (STS) en lugar de utilizar el manejador de seguridad. De esta manera, tiene la flexibilidad de proporcionar un proceso de señales más avanzado que el proceso ofrecido por el manejador de seguridad.

La interfaz de cliente de confianza es una mejora al programa proporcionado por CICS DFHPIRT. Este programa se utiliza para iniciar una interconexión cuando no se ha desplegado la aplicación de solicitante de servicio web utilizando el asistente de servicios web de CICS. Pero también puede actuar como la interfaz de cliente de confianza para STS.

Puede invocar la interfaz de cliente de confianza enlazando a DFHPIRT desde un manejador de mensajes o programa de proceso de cabeceras, pasando un canal denominado DFHWSTC-V1 y un conjunto de contenedores de seguridad. Utilizando estos contenedores, tiene la flexibilidad de solicitar una acción de emisión o validación desde el STS, seleccionar qué tipo de señal intercambiar y pasar la señal adecuada desde la cabecera de mensaje. DFHPIRT crea dinámicamente una interconexión, compone una solicitud de servicio web desde los contenedores de seguridad y la envía a STS.

DFHPIRT espera la respuesta del STS y la pasa de vuelta en el contenedor DFHWS-RESTOKEN al manejador de mensajes. Si el STS encuentra un error, devuelve un error SOAP. DFHPIRT pone el error en el contenedor DFHWS-STSFault y vuelve al programa de enlace en la interconexión.

Puede utilizar la interfaz de cliente de confianza sin habilitar el manejador de seguridad en las interconexiones del proveedor de servicios u del solicitante de servicio, o bien puede utilizar la interfaz de cliente de confianza además del manejador de seguridad.

Firma de mensajes SOAP

Para mensajes de entrada, CICS soporta firmas digitales en elementos en el cuerpo SOAP y en los bloques de cabecera SOAP. Para mensajes de salida, CICS firma todos los elementos en el cuerpo SOAP.

Un mensaje SOAP es un documento XML, que consta de un elemento <Envelope>, que contiene un elemento <Header> opcional y un elemento <Body> obligatorio.

La especificación WSS: SOAP Message Security permite que el contenido de <Header> y <Body> se firme a nivel de elemento. Es decir, en un mensaje determinado, los elementos individuales se pueden firmar o no, o se pueden firmar con firmas diferentes o utilizando algoritmos diferentes. Por ejemplo, en un mensaje SOAP utilizado en una aplicación de compra en línea, es adecuado firmar elementos que confirman el recibo de un pedido, porque estos elementos pueden tener un estado legal Sin embargo, para evitar la sobrecarga de firmas en un mensaje entero, otra información puede permanecer segura sin firmar.

Para mensajes de entrada, el manejador de mensajes de seguridad puede verificar una firma digital en elementos individuales en <Header> y <Body> de SOAP:

- Elementos firmados que se encuentran en <Header>.
- Elementos firmados en <Body> de SOAP. Si el manejador está configurado para esperar un cuerpo firmado, CICS rechaza cualquier mensaje SOAP en el que el cuerpo no está firmado y emite un error SOAP.

Para los mensajes de entrada, el manejador de mensajes de seguridad puede firmar el <Body> de SOAP únicamente; no firma <Header>. El algoritmo y la clave utilizados para firmar el cuerpo se especifican en la información de configuración del manejador.

Algoritmos de firma

CICS soporta los algoritmos de firma que requiere la especificación XML Signature. Cada algoritmo se identifica por medio de un identificador de recurso universal (URI).

Algoritmo	URI
Algoritmo de firma digital con Algoritmo de hash seguro 1 (DSA con SHA1) Soportado únicamente en los mensajes SOAP de entrada.	http://www.w3.org/2000/09/xmldsig#dsa-sha1
Algoritmo Rivest-Shamir-Adleman con Algoritmo de hash seguro 1 (RSA con SHA1)	http://www.w3.org/2000/09/xmldsig#rsa-sha1

Ejemplo de un mensaje SOAP firmado

Este ejemplo muestra un mensaje SOAP que se ha firmado por CICS.

1. La señal de seguridad binaria contiene la codificación base64binary del certificado X.509. Esta codificación incluye la clave pública que el destinatario deseado de mensaje SOAP utiliza para verificar la firma.
2. El algoritmo que se utiliza durante el proceso de hashing crea el resumen del mensaje.
3. El valor del resumen de mensaje.
4. El valor de resumen se cifra con la clave privada del usuario y se incluye aquí como valor de firma.
5. Hace referencia a la señal de seguridad binaria que contiene la clave pública que se utiliza para verificar la firma.

Para mensajes de entrada, CICS puede descifrar cualquier elemento cifrado en el cuerpo SOAP, y los bloques de cabecera SOAP cifrados donde el cuerpo también se cifra. Para mensajes de salida, CICS cifra todo el cuerpo SOAP.

Capítulo 4. Soporte para proteger los servicios web 613

La especificación WSS: *SOAP Message Security* permite que parte del contenido del elemento <Header> y todo el contenido del elemento <Body> se cifre a nivel de elemento. Es decir, en un mensaje determinado, los elementos individuales pueden tener diferentes niveles de cifrado, o se pueden cifrar utilizando diferentes algoritmos. Por ejemplo, en un mensaje SOAP utilizado en una aplicación de compra en línea, es adecuado cifrar los detalles de una tarjeta de crédito de un individuo para garantizar que siguen siendo confidenciales. Sin embargo, para evitar la sobrecarga de cifrado de todo el mensaje, parte de la información puede cifrarse de forma segura utilizando un algoritmo menos seguro (pero más rápido) y otra información puede permanecer segura sin cifrar.

Para mensajes de entrada, el manejador de mensajes de seguridad proporcionado por CICS puede descifrar elementos individuales en el <Body> SOAP, y puede descifrar elementos en la <Header> SOAP si el cuerpo SOAP también está cifrado. El manejador de mensajes de seguridad siempre descifra estos elementos:

- Elementos que encuentra en el elemento <Header> en el orden en el que se han encontrado los elementos.
- Elementos en el elemento <Body> de SOAP. Si desea rechazar un mensaje SOAP que no tiene un <Body> cifrado, configure el manejador para esperar un cuerpo cifrado utilizando el elemento <expect_encrypted_body>.

Para mensajes de salida, el manejador de mensajes de seguridad soporta el cifrado únicamente del contenido del <Body> SOAP; no cifra ningún elemento del elemento <Header>. Cuando el manejador de mensajes de seguridad cifra el elemento <Body>, todos los elementos del cuerpo se cifran con el mismo algoritmo y utilizando la misma clave. El algoritmo, y la información sobre la clave, se especifican en la información de configuración sobre el manejador.

Algoritmos de cifrado

CICS soporta los algoritmos de cifrado requeridos por la especificación de cifrado XML. Cada algoritmo se identifica por medio de un identificador de recurso universal (URI).

Algoritmo	URI
Algoritmo Triple Data Encryption Standard (Triple DES)	http://www.w3.org/2001/04/xmlenc#tripledes-cbc
Algoritmo Advanced Encryption Standard (AES) con una longitud de clave de 128 bits	http://www.w3.org/2001/04/xmlenc#aes128-cbc
Algoritmo Advanced Encryption Standard (AES) con una longitud de clave de 192 bits	http://www.w3.org/2001/04/xmlenc#aes192-cbc
Algoritmo Advanced Encryption Standard (AES) con una longitud de clave de 256 bits	http://www.w3.org/2001/04/xmlenc#aes256-cbc

Ejemplo de un mensaje SOAP cifrado

Este ejemplo de mensaje SOAP se ha cifrado mediante CICS.

```

<?xml version="1.0" encoding="UTF8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header>
  <wsse:Security xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    xmlns:wss="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
    xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
    xmlns:xenc="http://www.w3.org/2001/04/xmenc#" SOAP-ENV:mustUnderstand="1">

    <wsse:BinarySecurityToken
      EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary" 1
      ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509"
      wsu:Id="x509cert00">MIIChDCCAe2gAwIBAgIBADANBgkqhkiG9w0BAQUFADAwMQswCQYDVQQGEwJHJEMMAoGA1UEChMD
      SUJNMRMwEQYDVQQDEwpxaWxsIFlhdGVzMB4XDTA2MDEzMTAwMDAwMfoXDTA3MDEzMTIzNTk1OVow
      MDELMAkGA1UEBhMCR0IxDDAKBgNVBAoTA01CTTETMBEGA1UEAxMKV21sbCBZYXR1czCBnzANBgkq
      hkiG9w0BAQEFAAOBjQAwgYkCgYEArsRj/n+3RN75+jaxu0MBWShvZCB0egv8qu2UwLWEiogePsR
      6Ku4SuHbBwJtWNr0xBTAA591Ea70yhVdppx0nJB0CiERg7S0HudP7a8JXPfZA+BqV63JqRgJyxN6
      msfTAVEMR07LIxmZate62nwcFrvCKNPCFIJ5mkaJ9v1p7jkAwEAAa0BrTCBqJA/BgIghkgBhvHc
      AQ0EMhMwR2VuZjhdGKvIGJ5IHROZSBTZW1cm10eSBTZjJ2ZXIgzM9yIHovT1MgKFJBQ0YpMDgG
      A1UdEQQxMC+BEVdZQVRFU0BVSy5JQk0uQ09NggdJQk0uQ09NghtXV1cuSUJNLkNPTYcECRR1BjA0
      BgNVHQ8BAf8EBAMCAfYwHQYDVR00BBYEFmIPX6VZKP5+mSOY1TLNQGVJzu+MA0GCSqGSIb3DQEB
      BQUAA4GBAHdrS409Jhoe67pHL2gs7x4SpV/N0uJnn/w25sjjop3RLgJ2bKtK6RiEevhCDim6tnYW
      NyjBL1VdN7u5M6kTfd+HutR/HnIrQ3qPkXZK4ipgC0RWDJ+8APLySxtFL+J0LN9E06yjiHL68mq
      uZbTH2LvzFMypEqEbmkVbA87a1F

    </wsse:BinarySecurityToken>
    <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmenc#">
      <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmenc#rsa-1_5"/> 2
      <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <wsse:SecurityTokenReference>
          <wsse:Reference URI="#x509cert00"
            ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509"/> 3
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
        <xenc:CipherData>
          <xenc:CipherValue>M6bDQJtrvX0pEjAEIcf6bq6MP3ySmB4TQ0a/B5U1Qj1vWjD56V+GRJbF7ZCES5ojwCJHRVKW1ZB5 4
            Mb+aUzSW1soH2HQixclJchgWciYIn+E2TbG3R9m0zHD3XQsKtYVaOT1R7VPoMBd1ZLN0IomxjZn2
            p7JfxywXk0bcSLhdZnc=</xenc:CipherValue>
          </xenc:CipherData>
          <xenc:ReferenceList>
            <xenc:DataReference URI="#Enc1"/>
          </xenc:ReferenceList>
        </xenc:EncryptedKey>
      </wsse:Security>
    </SOAP-ENV:Header>
    <SOAP-ENV:Body>
      <xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmenc#" Id="Enc1" Type="http://www.w3.org/2001/04/xmenc#Content">
        <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmenc#tripleDES-cbc"/> 5
        <xenc:CipherData>
          <xenc:CipherValue>kgvqKnMcgiUn7r1lvkFXF0g4SodEd3dxAJo/mVN6ef211B1MZe1g70yJEHf4ZXw1Cdt0FebId1nK 6
            rrrksq11Mpw6So7ID8zav+KPQUKGm4+E=</xenc:CipherValue>
          </xenc:CipherData>
        </xenc:EncryptedData>
      </SOAP-ENV:Body>
    </SOAP-ENV:Envelope>

```

1. La señal de seguridad binaria contiene la codificación base64binary del certificado X.509. Esta codificación incluye la clave pública que se ha utilizado para cifrar la clave simétrica.
2. Indica el algoritmo que se ha utilizado para cifrar la clave simétrica.
3. Hace referencia a la señal de seguridad binaria que contiene la clave pública utilizada para cifrar la clave simétrica.
4. La clave simétrica cifrada que se ha utilizado para cifrar el mensaje.
5. El algoritmo de cifrado que se ha utilizado para cifrar el mensaje.
6. El mensaje cifrado.

Configuración de RACF para la Seguridad de servicios web

Debe configurar un gestor de seguridad externo, como RACF, para crear pares de clave pública-privada y certificados X.509 para firmar y cifrar mensajes SOAP de salida y para autenticar y descifrar mensajes SOAP de entrada firmados y cifrados.

Antes de empezar

Antes de realizar esta tarea, debe tener configurado RACF para trabajar con CICS. Especifique los parámetros de inicialización del sistema **DFLTUSER**, **KEYRING** y **SEC=YES** en la región de CICS que contiene las interconexiones de servicios web.

Nota: No se soportan varios certificados con el mismo nombre distinguido en el mismo **KEYRING**.

Procedimiento

1. Para autenticar mensajes SOAP de entrada que están firmados:
 - a. Importe el certificado X.509 en RACF como una clave ICSF.
 - b. Adjunte el certificado al conjunto de claves especificado en el parámetro de inicialización del sistema **KEYRING**, utilizando el mandato **RADCERT**:

```
RADCERT ID(userid1)
CONNECT(ID(userid2) LABEL('label-name') RING(ring-name))
```

donde:

- *userid1* es el ID de usuario predeterminado del conjunto de claves o tiene autoridad para conectar certificados al conjunto de claves para otros ID de usuario.
 - *userid2* es el ID de usuario que desea asociar al certificado.
 - *label-name* es el nombre del certificado.
 - *ring-name* es el nombre del conjunto de claves que se especifica en el parámetro de inicialización del sistema **KEYRING**.
- c. Opcional: Si desea utilizar identidades certificadas, asegúrese de que el ID de usuario asociado al certificado tiene autoridad subrogada para permitir que el trabajo se ejecute bajo otros ID de usuario. Además, asegúrese de que cualquier certificado adicional incluido en la cabecera de mensaje SOAP se importa a RACF.

El mensaje SOAP puede contener una señal de seguridad binaria en la cabecera que incluye el certificado o contiene una referencia al certificado. Esta referencia puede ser KEYNAME (la etiqueta de certificado en RACF), una combinación de ISSUER y el número de SERIAL o el SubjectKeyIdentifier. CICS puede reconocer SubjectKeyIdentifier sólo si se ha especificado como un atributo en la definición del certificado en RACF.

2. Para firmar mensajes SOAP de salida:
 - a. Cree un certificado X.509 y un par de claves pública-privada utilizando el mandato **RADCERT** siguiente:

```
RADCERT ID(userid2) GENCERT
SUBJECTSDN(CN('common-name')
            T('title')
            OU('organizational-unit')
            O('organization')
            L('locality')
            SP('state-or-province')
            C('country'))
WITHLABEL('label-name')
```

donde *userid2* es el ID de usuario que desea asociar al certificado. Cuando especifica el valor *label-name*, no utilice los siguientes caracteres:

< > ! =

- b. Adjunte el certificado al conjunto de claves especificado en el parámetro de inicialización del sistema **KEYRING**. Utilice el mandato **RADCERT**.
- c. Exporte el certificado y publíquelo en el destinatario deseado del mensaje SOAP.

Puede editar el archivo de configuración de interconexión para que CICS incluya automáticamente el certificado X.509 en la señal de seguridad binaria de la cabecera de mensaje SOAP para que el destinatario deseado valide la firma.

3. Para descifrar los mensajes SOAP de entrada que están cifrados, el mensaje SOAP debe incluir la clave pública que es parte de un par de claves donde la clave privada se define en CICS.
 - a. Genere un par de claves pública-privada y un certificado en RACF para el cifrado. El par de claves y el certificado deben generarse utilizando ICSF.
 - b. Adjunte el certificado al conjunto de claves especificado en el parámetro de inicialización del sistema **KEYRING**. Utilice el mandato **RACDCERT**.
 - c. Exporte el certificado y publíquelo en el generador de los mensajes SOAP que desea descifrar.

El generador del mensaje SOAP puede importar el certificado que contiene la clave pública y utilizarlo para cifrar el mensaje SOAP. El mensaje SOAP puede contener una señal de seguridad binaria en la cabecera que incluye la clave pública o contiene una referencia a ella. Esta referencia puede ser el **KEYNAME**, una combinación de **ISSUER** y el número de **SERIAL** o el **SubjectKeyIdentifier**. CICS puede reconocer el **SubjectKeyIdentifier** sólo si se ha especificado como un atributo en la definición de clave pública en RACF.

4. Para cifrar mensajes SOAP de salida:
 - a. Importe el certificado que contiene la clave pública que desea utilizar para el cifrado en RACF como una clave ICSF. El destinatario deseado debe tener la clave privada asociada a la clave pública para descifrar el mensaje SOAP.
 - b. Adjunte el certificado que contiene la clave pública al conjunto de claves especificado en el parámetro de inicialización del sistema **KEYRING**. Utilice el mandato **RACDCERT**.

CICS utiliza la clave pública en el certificado para cifrar el cuerpo SOAP y enviar el certificado que contiene la clave pública como una señal de seguridad binaria en la cabecera de mensaje SOAP. La clave pública se define en el archivo de configuración de interconexión.

Qué hacer a continuación

Esta configuración para firmar y cifrar mensajes de salida requiere que el certificado utilizado sea propiedad del ID de usuario de la región de CICS. El certificado tiene que ser propiedad del ID de usuario de la región de CICS porque RACF sólo permite al propietario del certificado extraer la clave privada, que se utiliza para el proceso de firma o cifrado.

Si CICS tiene que firmar o cifrar un mensaje utilizando un certificado que no posee, puede compartir un único certificado entre sistemas CICS siguiendo las instrucciones en .

Configuración de los servicios web de modalidad de proveedor para la propagación de identidad

La propagación de identidad con una solicitud de servicio web se basa en configuraciones basadas en la confianza; por ejemplo, utilizando una conexión SSL certificada por el cliente de WebSphere DataPower. En esta tarea, configure un recurso **PIPELINE** para esperar una señal de identidad **ICRX** en la cabecera de **WS-Security**, enviada desde un cliente de confianza.

Antes de empezar

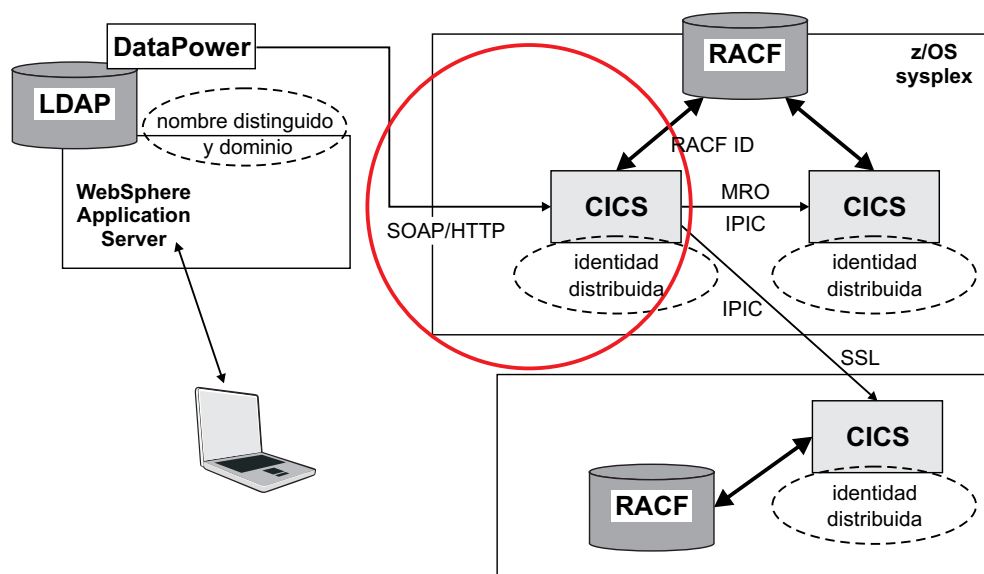
Debe configurar los valores de RACF RACMAP antes de configurar las conexiones de servicio web, de lo contrario, recibe el mensaje de RACF ICH408I para todas las solicitudes no correlacionadas que se envían a RACF. Para obtener más información sobre la configuración del mandato de RACF **RACMAP**, consulte Configuración de RACF para la propagación de identidad.

Debe configurar una relación de *confianza* entre el dispositivo WebSphere DataPower y CICS, por ejemplo, utilizando la certificación de cliente SSL entre WebSphere DataPower y CICS. El certificado digital que utiliza WebSphere DataPower para identificarse debe estar asociado a un ID de usuario, y ese ID de usuario debe tener autoridad sustituta para certificar identidades. Para obtener más información sobre la autoridad subrogada, consulte Seguridad de usuario subrogado.

Acerca de esta tarea

Esta tarea explica cómo utilizar CICS con un dispositivo WebSphere DataPower para proporcionar una configuración de servicio web que puede propagar identidades distribuidas de una forma sólida y segura. El círculo en el diagrama indica que esta tarea explica la configuración específica de CICS.

Figura 29. Configuración de CICS para esperar una señal de identidad ICRX de WebSphere DataPower.



WebSphere DataPower actúa como un intermediario entre CICS y otras aplicaciones. Las aplicaciones de solicitante de servicio web remoto se conectan al dispositivo WebSphere DataPower utilizando el protocolo SOAP. WebSphere DataPower autentica las credenciales proporcionadas por el cliente remoto y correlaciona las credenciales con una señal de identidad z/OS ICRX, que identifica la identidad distribuida de un usuario. El mensaje SOAP se reenvía a CICS a través de la conexión SSL de confianza con una señal de identidad ICRX en una

cabecera de WS-Security. Para obtener más información sobre las señales de identidad ICRX, consulte z/OS Security Server RACF Data Areas.

CICS recibe el mensaje SOAP de WebSphere DataPower. El archivo de configuración de PIPELINE especifica confianza *ciega*, porque el único cliente posible es el dispositivo WebSphere DataPower, y WebSphere DataPower se está comunicando con CICS a través de una conexión SSL segura. Por lo tanto, no necesita especificar una autenticación adicional en el archivo de configuración de PIPELINE. El programa de manejador de WS-Security ubica el primer ICRX encontrado en la cabecera de WS-Security y utiliza el ICRX para identificar el usuario.

Procedimiento

1. Cree un recurso PIPELINE, o edite un recurso PIPELINE existente para especificar la modalidad de ICRX básica que permite a PIPELINE recibir un ICRX. La combinación más típica es la confianza ciega con la modalidad ICRX básica. Para obtener más información sobre el elemento de recurso PIPELINE, consulte The <authentication> element.

Aquí tienen un archivo de configuración de PIPELINE de ejemplo, que muestra la confianza ciega con la modalidad ICRX básica:

```
<?xml version="1.0" encoding="EBCDIC-CP-US"?>
<provider_pipeline xmlns="http://www.ibm.com/software/http/cics/pipeline">
  <service>
    <service_handler_list>
      <wsse_handler>
        <dfhwsse_configuration version="1">
          <authentication trust="blind" mode="basic-ICRX"/>
        </dfhwsse_configuration>
      </wsse_handler>
    </service_handler_list>
    <terminal_handler>
      <cics_soap_1.2_handler/>
    </terminal_handler>
  </service>
  <apphandler>DFHPITP</apphandler>
</provider_pipeline>
```

Aquí tiene un mensaje SOAP de ejemplo con una identidad ICRX, utilizando la confianza ciega:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    <wsse:Security
      xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
      SOAP-ENV:mustUnderstand="1">

      <wsse:BinarySecurityToken EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary"
        wsu:Id="ICRX"
        xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wsswssecurity-utility-1.0.xsd"
        ValueType="http://www.ibm.com/xmlns/prod/zos/saf#ICRXV1">

          ICRX IS HERE

        </wsse:BinarySecurityToken>

      </wsse:Security>
    </SOAP-ENV:Header>
  </SOAP-ENV:Body>
```

APPLICATION SPECIFIC XML IS HERE

```
</SOAP-ENV:Body>  
</SOAP-ENV:Envelope>
```

2. Asegúrese de que WebSphere DataPower está configurado para poder enviar información de ICRX. Consulte Topologías de red de ejemplo para utilizar la propagación de identidad.

Resultados

Las solicitudes de servicio web de WebSphere DataPower con una señal de identidad ICRX en la cabecera de WS-Security, conectadas a través de una conexión SSL certificada por el cliente, puede fluir ahora.

Configuración de la interconexión para la Seguridad de servicios web

Para configurar una interconexión para soportar la Seguridad de servicios web (WSS), debe añadir un manejador de seguridad a los archivos de configuración de interconexión. Puede utilizar el manejador de seguridad proporcionado con CICS, tal y como se describe, o crear el suyo propio.

Antes de empezar

Antes de definir el manejador de seguridad proporcionado por CICS, debe identificar o crear los archivos de configuración de interconexión a los que añade la información de configuración para WSS.

Procedimiento

1. Añada un elemento `<wsse_handler>` a la interconexión. El manejador debe incluirse en el elemento `<service_handler_list>` en un proveedor de servicios o interconexión de solicitante. Codifique los siguientes elementos:

```
<wsse_handler>  
  <dfhwsse_configuration version="1">  
  
  </dfhwsse_configuration>  
</wsse_handler>
```

El elemento `<dfhwsse_configuration>` es un contenedor para los otros elementos de la configuración.

2. Opcional: Codifique un elemento `<authentication>`.
 - En una interconexión de solicitante de servicio, el elemento `<authentication>` especifica el tipo de autenticación que debe utilizarse en el manejador de seguridad de los mensajes SOAP de salida.
 - En una interconexión de proveedor de servicios, el elemento especifica si CICS utiliza señales de seguridad en un mensaje SOAP de entrada para determinar el ID de usuario bajo el que se procesa el trabajo.
 - a. Codifique el atributo **trust** para especificar si se utiliza la identidad afirmada y la naturaleza de la relación de confianza entre el solicitante y el proveedor de servicios. Para obtener detalles del atributo **trust**, consulte The `<authentication>` element.
 - b. Opcional: Si ha especificado **trust=none**, codifique el atributo **mode** para especificar cómo se procesan las credenciales encontradas en el mensaje. Para obtener detalles del atributo **mode**, consulte The `<authentication>` element.
 - c. En el elemento `<authentication>`, codifique estos elementos:

- 1) Un elemento <suppress/> vacío opcional.
 Si este elemento se especifica en una interconexión de proveedor de servicios, el manejador no intenta utilizar ninguna señal de seguridad en el mensaje para determinar bajo qué ID de usuario se ejecuta el trabajo.
 Si este elemento se especifica en una interconexión de solicitante de servicio, el manejador no intenta añadir al mensaje SOAP de salida ninguna señal de seguridad necesaria para la autenticación.
 - 2) En una interconexión de solicitante, un elemento <algorithm> opcional que especifica el URI del algoritmo que se utiliza para firmar el cuerpo del mensaje SOAP. Debe especificar este elemento si la combinación de los valores de los atributos trust y mode indican que los mensajes están firmados. Sólo puede especificar el RSA con el algoritmo SHA1 en este elemento. El URI es <http://www.w3.org/2000/09/xmldsig#rsa-sha1>.
 - 3) Un elemento <certificate_label> opcional que especifica la etiqueta asociada a un certificado digital X.509 instalado en RACF. Si especifica este elemento en una interconexión de solicitante de servicio y el elemento <suppress> no se especifica, el certificado se añade a la cabecera de seguridad en el mensaje SOAP. Si no especifica un elemento <certificate_label>, CICS utiliza el certificado predeterminado en el anillo de claves RACF.
 Este elemento se ignora en una interconexión de proveedor de servicios.
3. Opcional: Codifique un elemento <sts_authentication> como alternativa al elemento <authentication>. No debe codificar ambos en el archivo de configuración de interconexión. Este elemento especifica que el Servicio de token de seguridad (STS) se utiliza para autenticación y determina el tipo de solicitud que se envía.
 - a. Opcional: Únicamente en la modalidad de proveedor de servicios, codifique el atributo **action** para especificar si el STS verifica o intercambia una señal de seguridad. Para obtener detalles del atributo **action**, consulte The <sts_authentication> element.
 - b. En el elemento <sts_authentication>, codifique estos elementos:
 - 1) Un elemento <auth_token_type>. Este elemento es obligatorio cuando especifica un elemento <sts_authentication> en una interconexión de solicitante de servicio y es opcional en un proveedor de servicios. Para obtener más información, consulte <auth_token_type>.
 - En una interconexión de solicitante de servicio, el elemento <auth_token_type> indica el tipo de señal que emite STS cuando CICS la envía al ID de usuario contenido en el contenedor DFHWS-USERID. La señal que recibe CICS desde el STS se coloca en la cabecera del mensaje de salida.
 - En una interconexión de proveedor de servicios, el elemento <auth_token_type> se utiliza para determinar la señal de identidad que toma CICS de la cabecera de mensaje y la envía al STS para intercambiar o validar. CICS utiliza la primera señal de identidad del tipo especificado en la cabecera de mensaje. Si no especifica este elemento, CICS utiliza la primera señal de identidad que se encuentra en la cabecera de mensaje. CICS no considera los siguientes elementos como señales de identidad:
 - wsu:Timestamp
 - xenc:ReferenceList

- xenc:EncryptedKey
 - ds:Signature
- 2) En una interconexión de proveedor de servicios solamente, un elemento `<suppress/>` vacío opcional. Si se especifica este elemento, el manejador no intenta utilizar señales de seguridad en el mensaje para determinar el ID de usuario bajo el que se ejecuta el trabajo. El elemento `<suppress/>` incluye la señal de identidad que devuelve STS.
4. Opcional: Codifique un elemento `<sts_endpoint>`. Utilice este elemento sólo si ya ha especificado un elemento `<sts_authentication>`. En el elemento `<sts_endpoint>`, codifique los siguientes elementos:
- Un elemento `<endpoint>`. Este elemento contiene un URI que apunta a la ubicación del servicio de señales de seguridad (STS) en la red. Se recomienda que utilice SSL o TLS para mantener la conexión al STS segura, en lugar de utilizar HTTP.
Para utilizar el soporte SAML, establezca el punto final en `cics://PROGRAM/DFHSAML`.
También puede especificar un punto final WebSphere MQ, utilizando el formato JMS de URI.
 - Un elemento `<jvmserver>` opcional. Este elemento identifica el servidor JVM configurado para ejecutar el servicio de señal SAML. Si este elemento no está incluido, el servidor JVM del recurso de muestra predeterminado DFHXSTS se asume. Este elemento es válido sólo si está utilizando SAML: si lo utiliza en otras situaciones, se produce un error
5. Opcional: Si necesita que los mensajes SOAP de entrada se firmen digitalmente, codifique un elemento `<expect_signed_body/>` vacío.
El elemento `<expect_signed_body/>` indica que el cuerpo (`<body>`) del mensaje de entrada debe estar firmado. Si el cuerpo del mensaje de entrada no está correctamente firmado, CICS rechaza el mensaje con un error de seguridad.
6. Opcional: Si desea rechazar los mensajes SOAP de entrada que están firmados digitalmente, codifique un elemento `<reject_signature/>` vacío.
7. Opcional: Si necesita que los mensajes SOAP de entrada estén cifrados, codifique un elemento `<expect_encrypted_body/>` vacío.
El elemento `<expect_encrypted_body/>` indica que el `<body>` del mensaje de entrada debe estar cifrado. Si el cuerpo del mensaje de entrada no está correctamente cifrado, CICS rechaza el mensaje con un error de seguridad.
8. Si desea rechazar los mensajes SOAP de entrada que están cifrados parcial o completamente, codifique un elemento `<reject_encryption/>` vacío.
9. Opcional: Si necesita que los mensajes SOAP de entrada estén firmados, codifique un elemento `<sign_body>`.
- a. En el elemento `<sign_body>`, codifique un elemento `<algorithm>`.
 - b. Después del elemento `<algorithm>`, codifique un elemento `<certificate_label>`.
- Aquí tiene un ejemplo de un elemento `<sign_body>` completado:
- ```
<sign_body>
 <algorithm>http://www.w3.org/2000/09/xmldsig#rsa-sha1</algorithm>
 <certificate_label>SIGCERT01</certificate_label>
</sign_body>
```
10. Opcional: Si necesita que los mensajes SOAP de salida estén cifrados, codifique un elemento `<encrypt_body>`.
- a. En el elemento `<encrypt_body>`, codifique un elemento `<algorithm>`.

- b. Después del elemento `<algorithm>`, codifique un elemento `<certificate_label>`.

Aquí tiene un ejemplo de un elemento `<encrypt_body>` completado:

```
<encrypt_body>
 <algorithm>http://www.w3.org/2001/04/xmlenc#tripledes-cbc</algorithm>
 <certificate_label>ENCCERT02</certificate_label>
</encrypt_body>
```

## Ejemplo

El siguiente ejemplo muestra un manejador de seguridad completado en el que están presentes la mayoría de los elementos opcionales:

```
<wsse_handler>
 <dfhwsse_configuration version="1">
 <authentication trust="signature" mode="basic">
 <suppress/>
 <certificate_label>AUTHCERT03</certificate_label>
 </authentication>
 <expect_signed_body/>
 <expect_encrypted_body/>
 <sign_body>
 <algorithm>http://www.w3.org/2000/09/xmldsig#rsa-sha1</algorithm>
 <certificate_label>SIGCERT01</certificate_label>
 </sign_body>
 <encrypt_body>
 <algorithm>http://www.w3.org/2001/04/xmlenc#tripledes-cbc</algorithm>
 <certificate_label>ENCCERT02</certificate_label>
 </encrypt_body>
 </dfhwsse_configuration>
</wsse_handler>
```

---

## Escritura de un manejador de seguridad personalizado

Si desea utilizar su propio proceso y procedimientos de seguridad, puede escribir un manejador de mensajes personalizado para procesar mensajes SOAP seguros en la interconexión.

### Antes de empezar

Debe decidir el nivel de seguridad que debe soportar su manejador de seguridad, y asegurarse de que se devuelve un error SOAP adecuado cuando el mensaje incluye seguridad que no se soporta.

### Acerca de esta tarea

El manejador de mensajes también tiene que poder afrontar la seguridad en mensajes de entrada y de salida.

### Procedimiento

1. Recupere el contenedor DFHREQUEST o DFHRESPONSE utilizando un mandato **EXEC CICS GET CONTAINER**.
2. Analice el XML para encontrar la señal de seguridad que está en la cabecera de mensaje de WS-Security. La cabecera comienza con el elemento `<wsse:Security>`. La señal de seguridad puede ser un nombre de usuario y contraseña, un certificado digital o una clave de cifrado. Un mensaje puede tener muchas señales en la cabecera de seguridad, por lo que su manejador necesita identificar la correcta para procesarla.

3. Lleve a cabo el proceso adecuado, dependiendo de la seguridad que se implementa en el mensaje.
  - a. Si desea realizar una autenticación básica de una señal Kerberos, emita un mandato **EXEC CICS VERIFY TOKEN**. Este mandato comprueba que la señal Kerberos proporcionada es válida. Si el mandato es correcto, actualice el contenedor DFHWS-USERID con un **EXEC CICS PUT CONTAINER**. De lo contrario, emita un mandato **EXEC CICS SOAPFAULT CREATE**.
  - b. Si desea realizar autenticación básica de una contraseña o frase de contraseña, emita un mandato **EXEC CICS VERIFY PHRASE**. Este mandato comprueba el nombre de usuario y la contraseña en la cabecera de seguridad del mensaje. Si el mandato es correcto, actualice el contenedor DFHWS-USERID con un **EXEC CICS PUT CONTAINER**. De lo contrario, emita un mandato **EXEC CICS SOAPFAULT CREATE**.
  - c. Si desea realizar una autenticación avanzada, ya sea intercambiando o validando un rango de señales con un Security Token Service, utilice la interfaz de cliente de confianza. Consulte “Invocación del cliente de confianza desde un manejador de mensajes” para obtener más detalles.
  - d. Valide las credenciales del certificado digital si el mensaje está firmado.
  - e. Si hay partes del mensaje cifradas, descifre el mensaje utilizando la información en la cabecera de seguridad. La especificación How CICS complies with Web Services Security specifications proporciona información sobre cómo hacer esto.

## Resultados

Defina el programa de manejador de seguridad en CICS y actualice el archivo de configuración de interconexión, garantizando que está colocado correctamente en el XML. En un archivo de configuración de interconexión de solicitante de servicio, el manejador de seguridad debe configurarse para ejecutarse al final de la interconexión. En un archivo de configuración de interconexión de proveedor de servicios, el manejador de seguridad debe configurarse para ejecutarse al principio de la interconexión.

## Qué hacer a continuación

Para obtener información general sobre cómo escribir un manejador de mensajes personalizado, consulte la publicación de IBM Redbooks, *Application Development for CICS Web Services*, que está disponible en <http://www.redbooks.ibm.com/abstracts/sg247126.html>.

---

## Invocación del cliente de confianza desde un manejador de mensajes

CICS proporciona una interfaz para que pueda escribir su propio manejador de mensajes para invocar un Security Token Service (STS). Con esta interfaz, puede realizar un proceso más avanzado que el manejador de seguridad proporcionado por CICS.

### Antes de empezar

### Acerca de esta tarea

Puede utilizar el cliente de confianza en lugar del manejador de seguridad o de forma adicional con el mismo. Para utilizar la interfaz del cliente de confianza:



## Procedimiento

1. Extraiga la señal correcta de la cabecera del mensaje de seguridad del mensaje de entrada o de salida.
2. Enlace con el programa DFHPIRT, analizando el canal DFHWSTC-V1 y los siguientes contenedores necesarios:
  - DFHWS-STSUR, que contiene la ubicación del STS en la red.
  - DFHWS-STSACTION, que contiene el URI del tipo de solicitud que el STS debe realizar. Las dos acciones soportadas son `issue` y `validate`.
  - DFHWS-IDTOKEN, que contiene la señal que debe verificar o extraer el STS.
  - DFHWS-TOKENTYPE, que contiene el tipo de señal que el STS debe enviar de vuelta en la respuesta.
  - DFHWS-SERVICEURI, que contiene el URI de la operación de servicio web que se está invocando.

De manera opcional, puede incluir el contenedor DFHWS-XMLNS para proporcionar los espacios de nombres del mensaje SOAP que contiene la señal de seguridad. Este contenedor se describe con más detalles en The header processing program interface.

3. DFHPIRT se devuelve con la respuesta del STS. Una respuesta correcta se devuelve en el contenedor DFHWS-RESTOKEN.

Si el STS se encuentra un problema con la solicitud, devuelve un error SOAP. DFHPIRT coloca el error SOAP en el contenedor DFHWS-STSFault. Si el STS proporciona un motivo para emitir el error SOAP, el motivo se coloca en el contenedor DFHWS-STReason.

Si se produce una terminación anómala, se devuelve un contenedor DFHError que contiene detalles del error de proceso.

El manejador de mensajes debe gestionar estas respuestas y realizar un proceso adecuado en caso de un error. Por ejemplo, el manejador de mensajes puede devolver un error SOAP adecuado al solicitante de servicio web.

4. Procese la respuesta como adecuada. En modalidad de proveedor, el proceso de interconexión debe garantizar que el nombre de usuario que CICS puede entender se coloca en el contenedor DFHWS-USERID en el momento en el que el mensaje alcanza el manejador de aplicación. En modalidad de solicitante, el manejador de mensajes debe añadir la señal correcta a la cabecera de seguridad del mensaje de salida.

## Qué hacer a continuación

Cuando haya escrito el manejador de mensajes, despliegue el programa en CICS y actualice los archivos de configuración de interconexión adecuados. En las interconexiones de solicitante de servicio, defina el manejador de mensajes al final del proceso de interconexión pero antes del manejador de seguridad proporcionado por CICS. En las interconexiones de proveedor de servicios, defina el manejador de mensajes al principio de la interconexión pero después del manejador de seguridad proporcionado por CICS.

---

## Seguridad para z/OS Connect

z/OS Connect es una aplicación WebSphere Liberty, y tiene la misma configuración y consideraciones que otras aplicaciones WebSphere Liberty. Además, z/OS Connect for CICS 1.0 y z/OS Connect Enterprise Edition tienen algunos requisitos de seguridad adicionales.

z/OS Connect for CICS 1.0 y z/OS Connect Enterprise Edition tienen una interfaz de gestión RESTful para permitir un descubrimiento de servicio dinámico. Esta interfaz está alojada en el mismo nombre de host y número de puerto que los servicios JSON individuales. Se recomienda el uso de la Seguridad de la capa de transporte (TLS) para proteger esta interfaz y los servicios JSON individuales.

De forma predeterminada, todas las conexiones de cliente para z/OS Connect for CICS 1.0 y z/OS Connect Enterprise Edition deben utilizar el protocolo HTTPS. El comportamiento predeterminado es requerir una conexión TLS certificada por el cliente para CICS. Si se retiene este valor predeterminado, los certificados de cliente deben estar asociados a un ID de usuario SAF. La aplicación se ejecuta utilizando esta identidad derivada del certificado.

z/OS Connect for CICS 1.0 y z/OS Connect Enterprise Edition se pueden configurar para soportar el protocolo de autenticación básica HTTP. Este protocolo permite a un cliente conectarse utilizando TLS junto con un ID de usuario de SAF y una contraseña. Para habilitar el soporte para la autenticación básica HTTP, añada la línea siguiente al archivo de configuración del servidor Liberty (server.xml) : <webAppSecurity allowFailOverToBasicAuth="true"/>

Los usuarios de z/OS Connect for CICS 1.0 y z/OS Connect Enterprise Edition deben ser miembro de zos.connect.access.roles.zosConnectAccess EJBROLE. Para obtener más información, consulte

Consulte Configuring authorization for applications in Liberty para obtener información de Liberty, Configuring security for z/OS Connect para obtener información de z/OS Connect y Configuring security for z/OS Connect EE para obtener información de z/OS Connect EE.

Para obtener más información, consulte y Configuring security for a Liberty JVM server.

## Configuración de permisos para APIs y servicios de z/OS Connect

El modelo de seguridad de CICS requiere acciones adicionales de forma que pueda configurar permisos para APIs y servicios con z/OS Connect for CICS 1.0 y z/OS Connect Enterprise Edition.

### Acerca de esta tarea

Cuando se utiliza z/OS Connect para inyectar trabajo en CICS, las dos identidades siguientes se asocian al trabajo en diferentes etapas del proceso:

- Se asigna una identidad temporal e inicial durante el proceso de adjuntar el trabajo.
- A continuación, se utiliza una identidad autenticada para ejecutar el resto del trabajo.

Puede configurar estas identidades de varias formas, dependiendo de sus preferencias y del entorno del sistema.

### Procedimiento

1. Opcional: Cree un ID de usuario inicial alternativo para z/OS Connect.

De forma predeterminada, la identidad inicial es el ID de usuario CICS predeterminado, pero puede elegir asignar un ID de usuario diferente para evitar darle al ID de usuario CICS predeterminado permiso para ejecutar la transacción CPIH, o su equivalente.

- a. Autorice al ID de usuario inicial alternativo para ejecutar la transacción CPIH y cualquier otra transacción iniciada a través de z/OS Connect.

El ID de usuario inicial requiere permiso para ejecutar la transacción de destino para la API o el servicio.

2. Asigne un ID de usuario inicial predeterminado. Puede elegir uno o los dos métodos siguientes:

- Establezca un valor de sustitución de ID de usuario en el perfil JVM para el recurso JVMSERVER que aloja z/OS Connect.

A continuación, se muestra una sustitución de ejemplo, donde ZOSCUSER es el ID de usuario inicial predeterminado:

```
-Dcom.ibm.cics.jvmserver.http.userid=ZOSCUSER
```

**Nota:** Si establece un ID de usuario inicial predeterminado en el perfil JVM, no necesita proporcionar un valor USERID para cada URIMAP. Sin embargo, si proporciona USERID para URIMAP y un valor de sustitución en el perfil JVM, el USERID especificado para un URIMAP determinado tiene preferencia.

- Establezca el campo USERID para un recurso URIMAP predeterminado que tiene como destino z/OS Connect.

Cuando z/OS Connect recibe una solicitud HTTP, CICS coincide con ella en los recursos URIMAP que están instalados. Si el URIMAP que se encuentra especifica el atributo USERID, se utiliza ese ID de usuario como ID de usuario inicial, en lugar del ID de usuario inicial predeterminado para el servidor JVM.

Aquí hay una configuración de ejemplo para un recurso URIMAP denominado ZOSCDEFT, donde JVMSERVER es el valor USAGE, se establece un valor genérico para el atributo PATH, CPIH es la transacción de destino y ZOSCUSER es el ID de usuario inicial predeterminado:

```
NAME: ZOSCDEFT
USAGE: JVMSERVER
SCHEME: HTTP
PORT: NO
HOST: *
PATH: /zosConnect/*
TRANSACTION: CPIH
USERID: ZOSCUSER
```

**Nota:** Es poco probable que los recursos URIMAP que se instalan utilizando el mecanismo PIPELINE SCAN se configuren con un ID de usuario predeterminado. En este escenario, puede tener en cuenta especificar un valor de sustitución de ID de usuario en JVMSERVER.

**Nota:** Es posible almacenar un ID de usuario inicial en un archivo WSBIND: el usuario de DFHLS2JS o DFHJS2LS puede proporcionar un valor para el parámetro de entrada **USERID**. Si se utiliza el parámetro **USERID**, cualquier URIMAPs creada durante una PIPELINE SCAN incluye el ID de usuario inicial solicitado.

## Resultados

Ahora tiene configurado el entorno así que CICS reconoce los URI para las APIs y servicios, y asocia un ID de usuario inicial para su uso cuando se conecta a la transacción de destino.

---

## Capítulo 5. Resolución de problemas de servicios web

Los problemas que pueda tener al implementar servicios web en CICS pueden producirse durante el proceso de despliegue o en el tiempo de ejecución cuando CICS está transformando los mensajes.

---

### Resolución de problemas de servicios web SOAP

Los problemas que pueda tener al implementar servicios web SOAP en CICS pueden producirse durante el proceso de despliegue o en el tiempo de ejecución cuando CICS está transformando los mensajes SOAP.

### Diagnóstico de errores de despliegue

Los errores de despliegue pueden producirse cuando intenta ejecutar los trabajos por lotes del asistente de servicios web de CICS o los trabajos por lotes del asistente XML de CICS, instalar un recurso PIPELINE en CICS o instalar un recurso WEBSERVICE en CICS. Aquí se describen los errores de despliegue más frecuentes, incluido el síntoma del problema, la causa y la solución.

#### Acerca de esta tarea

En el caso de un error de despliegue, los recursos PIPELINE se instalan por lo general en un estado DISABLED y los recursos WEBSERVICE se instalan en un estado UNUSABLE.

Los mensajes informativos y de error asociados con los trabajos por lotes del asistente de servicios web de CICS y los trabajos por lotes del asistente XML de CICS se ubican en el registro de trabajo. Los mensajes de error asociados con los recursos de instalación se encuentran en el registro del sistema.

Los asistentes emiten los códigos de 0, 4, 8 o 12, los otros códigos los suelen emitir BPXBATCH, JVM o IEBGENER.

Los códigos emitidos por BPXBATCH entran en dos categorías principales; un código de menos de 128 indica un error de mandato, un código mayor o igual que 128 indica que el proceso ha finalizado con una señal. Para obtener más información sobre BPXBATCH y sus códigos de retorno, consulte *z/OS UNIX System Services Command Reference*.

#### Procedimiento

- Recibe un código de retorno de 0, 4, 8 o 12 al ejecutar los trabajos por lotes del asistente de servicios web de CICS o los trabajos por lotes del asistente XML de CICS. Los códigos de retorno significan lo siguiente:
  - 0 - El trabajo se ha completado satisfactoriamente.
  - 4 - Aviso. El trabajo se ha completado correctamente, pero se han emitido uno o más mensajes de aviso.
  - 8 - Error de entrada. El trabajo no ha finalizado satisfactoriamente. Uno o más mensajes de error se han emitido durante la validación de los parámetros de entrada.
  - 12 - Error. El trabajo no ha finalizado satisfactoriamente. Uno o más mensajes de error se han emitido durante la ejecución.

1. Compruebe el registro de trabajos para ver los mensajes de error o de aviso. Busque las explicaciones detalladas de los mensajes. Las explicaciones por lo general describen las acciones que puede realizar para arreglar el problema.
  2. Asegúrese de que ha introducido valores correctos para cada uno de los parámetros del trabajo. Los valores de parámetro, como los nombres de archivo y los elementos de la descripción de servicio web, deben distinguirse entre mayúsculas y minúsculas.
  3. Asegúrese de que se ha especificado una combinación de parámetros correcta. Por ejemplo, si incluye el parámetro **PGMNAME** en DFHWS2LS al generar un archivo de enlace de servicio web para un solicitante de servicio, obtiene un error y el trabajo no se completa correctamente.
- Recibe un código de retorno de 1, 136 o 139 al ejecutar los trabajos por lotes del asistente de servicios web de CICS o los trabajos por lotes del asistente XML de CICS. Estos códigos de retorno indican que la JVM ha fallado, normalmente porque no tiene suficiente almacenamiento disponible. Los asistentes de CICS requieren un tamaño de la región de JCL de al menos 300 MB, aunque algunos documentos pueden requerir 400 MB.
    1. Aumente el tamaño de la región o tenga en cuenta establecer el tamaño de la región en 0M.
    2. Compruebe si hay alguna salida IEFUSI activa, que puede limitar el tamaño de la región.

**Nota:** Si utiliza un JVM de 64bit, asegúrese de que especifica un valor MEMLIMIT adecuado.

- Recibe un código de retorno de 137 al ejecutar el trabajo por lotes del asistente de servicios web de CICS, DFHLS2WS, o el trabajo por lotes del asistente XML de CICS, DFHLS2SC. Este código de retorno significa que el trabajo ha excedido el tiempo de espera.
  1. Aumente el tiempo codificando el parámetro **TIME** en la sentencia EXEC del trabajo en **TIME=1440**, o aumente el valor MAXCPU TIME en el miembro SYS1.PARMLIB(BPXPRMxx).
- Recibe un mensaje de error DFHPI0914 al intentar instalar un recurso WEBSERVICE. El mensaje incluye algo de información sobre la causa del error de instalación.
  1. Compruebe que ha otorgado permiso a CICS para leer el archivo de enlace de servicio web en z/OS UNIX.
  2. Compruebe que el archivo de enlace de servicio web no está dañado. Esto puede ocurrir, por ejemplo, si utiliza FTP para transferir el archivo a z/OS UNIX en modalidad de texto en lugar de modalidad binaria.
  3. Compruebe que los dos archivos de enlace de servicio web con el mismo nombre no estén en diferentes directorios de recogida.
  4. Si está intentando instalar un recurso para una aplicación de solicitante de servicio web, compruebe que la versión del enlace SOAP coincide con el nivel soportado en la interconexión. No puede instalar un WEBSERVICE SOAP 1.1 en una interconexión de solicitante de servicio que soporta SOAP 1.2.
  5. Compruebe que no esté instalando un recurso de WEBSERVICE en modo de proveedor en una interconexión en modo de solicitante. Los archivos de enlace de servicio web especifican un valor **PROGRAM**, en tanto que los archivos en modo de solicitante no.
  6. Si está utilizando DFHWS2LS o DFHLS2WS, compruebe que ha especificado los parámetros correctos al generar el archivo de enlace de servicio web.

Algunos parámetros, como **PGMNAME**, sólo se permiten para los proveedores de servicios web y se deben excluir si está creando un solicitante de servicio web.

7. Si va a utilizar DFHWS2LS o DFHLS2WS, compruebe los mensajes emitidos por el trabajo para ver si hay algún problema que sea necesario resolver antes de crear el recurso WEBSERVICE.
- El recurso PIPELINE no se instala y se recibe un mensaje de error DFHPI0700, DFHPI0712, DFHPI0714 o similar.
    1. Si ha recibido un mensaje de error DFHPI0700, debe habilitar el soporte de lenguaje PL/I en la región de CICS. Esto es necesario antes de poder instalar los recursos PIPELINE. Consulte Language Environment support for PL/I para obtener más información.
    2. Compruebe que ha otorgado autorización a CICS para acceder a los directorios de z/OS UNIX para leer los archivos de configuración de interconexión.
    3. Compruebe que el directorio que se especifica en el parámetro **wsdir** sea válido. En particular, compruebe que el directorio y los nombres de archivo en z/OS UNIX distingan entre mayúsculas y minúsculas.
    4. Compruebe que no tiene un recurso PIPELINE con el mismo nombre en un estado ENABLED en la región de CICS.
  - El recurso PIPELINE se instala en un estado DISABLED. Se obtiene un mensaje de error en el rango de DFHPI0702 a DFHPI0711.
    1. Compruebe que no haya errores en el archivo de configuración de interconexión. Los elementos del archivo de configuración de interconexión solo pueden aparecer en determinados lugares. Si especifica esto incorrectamente, obtiene un mensaje de error DFHPI0702. Este mensaje incluye el nombre del elemento que está causando el problema. Compruebe la descripción del elemento para asegurarse de que la ha codificado en el lugar correcto.
    2. Compruebe que no existan caracteres no imprimibles, como pestañas, en el archivo de configuración de la interconexión.
    3. Asegúrese de que el XML sea válido. Si el XML no es válido, puede originar errores de análisis al intentar instalar el recurso PIPELINE.
    4. Asegúrese de que el archivo de configuración de interconexión esté codificado en US EBCDIC. Si intenta utilizar una codificación EBCDIC diferente, CICS no podrá procesar el archivo.
  - El recurso WEBSERVICE está en un estado DISABLED. Los estados DISABLED y DISABLING sólo están disponibles para recursos WEBSERVICE definidos e instalados en paquetes CICS.
    1. Si el recurso PIPELINE asociado al recurso WEBSERVICE se ha descartado, el recurso WEBSERVICE entra en estado DISABLED. Investigue porque falta el recurso PIPELINE, y sustitúyalo si es necesario.
    2. Si se ha llevado a cabo una acción de inhabilitación para el paquete CICS donde está definido el recurso WEBSERVICE, el recurso WEBSERVICE entra en estado DISABLED cuando el servicio web ya no está en uso. Investigue el estado del paquete CICS y habilítelo si es necesario.

## Diagnóstico de errores de tiempo de ejecución del proveedor de servicios

Si está teniendo problemas al recibir o procesar mensajes de entrada en la interconexión de modalidad de proveedor, puede haber un problema con el transporte o un mensaje SOAP específico.

## Procedimiento

- Recibirá un mensaje DFHPI0401, DFHPI0502 o similar, indicando que se ha producido un error de transporte WebSphere MQ o HTTP. Si el transporte es HTTP, el cliente recibe un mensaje de error 500 Server Internal Error. Si el transporte es WebSphere MQ, el mensaje está escrito en la cola de mensajes no entregados (DLQ). No se devuelve un error SOAP al solicitante de servicio web porque CICS no puede determinar qué tipo de mensaje se ha recibido.
- Recibe un mensaje DFHxx y un mensaje de error 404 Not Found.
  1. Si no está utilizando el asistente de servicios web, debe crear un recurso URIMAP. Si está utilizando el asistente de servicios web, la URIMAP se crea automáticamente cuando ejecuta el mandato **PIPELINE SCAN**. El registro del sistema proporciona información sobre cualquier error que se ha producido como resultado de la ejecución de este mandato.
  2. Compruebe que el recurso WEBSERVICE está habilitado y que la URIMAP a la que está asociada es lo que esperaba. Si el recurso WEBSERVICE está en un estado UNUSABLE, consulte “Diagnóstico de errores de despliegue” en la página 629.
  3. Compruebe que ha especificado correctamente el URI y el número de puerto. En particular, compruebe si son mayúsculas o minúsculas porque el atributo PATH en el recurso URIMAP distingue entre mayúsculas y minúsculas.
- Si se están notificando errores inesperados, piense en utilizar CEDX para depurar la aplicación de servicio web.
  1. Compruebe el registro del sistema para ver qué mensajes de error está notificando CICS. Esto puede indicar qué tipo de error se está produciendo. Si CICS no está informando de ningún error, asegúrese de que la solicitud está llegando a CICS a través de la red.
  2. Ejecute CEDX en CPIH para el transporte HTTP, CPIQ para el transporte WebSphere MQ o la transacción que ha especificado en URIMAP si es diferente.

Si se produce un intercambio de tareas durante el proceso de interconexión antes del manejador de aplicación, a menos que el contenedor DFHWS-TRANID esté lleno, la nueva tarea se ejecuta bajo el mismo ID de transacción que la primera. Esto puede interferir con la ejecución de CEDX, porque la primera tarea tiene un bloqueo en la sesión CEDX. Puede evitar este problema utilizando DFHWS-TRANID para cambiar el ID de transacción cuando se conmuta la tarea, lo que le permite utilizar CEDX en las tareas de aplicación e interconexión por separado. Para obtener más información sobre CEDX, consulte.
  3. Si CEDX no se activa o no le permite resolver el problema, considere ejecutar un rastreo auxiliar con los dominios PI, SO, AP, EI y XS activos. Esto puede indicar si hay un problema de seguridad, un problema TCP/IP, un problema de programa de aplicación o un problema de interconexión en la región de CICS. Busque cualquier punto de rastreo de excepción o terminación anómala.
- Si está recibiendo errores de conversión, consulte “Diagnóstico de errores de conversión de datos” en la página 637.
- Si piensa que su problema está relacionado con los mensajes MTOM, consulte “Diagnóstico de errores de MTOM/XOP” en la página 635.
- Si una conexión HTTP permanente está cerrada periódicamente:
  1. Compruebe si el ajuste de rendimiento está habilitado para conexiones HTTP. Para obtener más información, consulte SOTUNING



2. Si el ajuste de rendimiento está habilitado, CICS cerrará periódicamente las conexiones HTTP permanentes para permitir que las conexiones se redistribuyan entre las regiones que están escuchando los puntos finales de IP compartida.
- Si se rechazan los intentos de conexión cuando la región de CICS está en su capacidad máxima.
  1. Compruebe si el ajuste de rendimiento está habilitado para conexiones HTTP. Para obtener más información, consulte SOTUNING
  2. Si el ajuste de rendimiento está habilitado, cuando CICS está en su capacidad máxima, todas las solicitudes abiertas de conexión HTTP de entrada harán cola fuera de CICS en la cola de pendientes de la conexión de escucha de TCPIP SERVICE en TCP/IP. El campo TCP/IP Global statistics SOG\_PAUSING\_HTTP\_LISTENING de la región refleja si es éste el caso actualmente, y los campos SOG\_TIMES\_AT\_ACCEPT\_LIMIT/ SOG\_TIME\_LAST\_PAUSED\_HTTP\_LISTENING contienen más información sobre apariciones anteriores.
  3. Utilice **NETSTAT ALL** para obtener la estadística ConnectionsDropped para la conexión de escucha de TCPIP SERVICE. Este es el número de solicitudes de conexión recibidas y descartadas porque el número de solicitudes de conexión que está esperando en la cola de pendientes ha alcanzado el límite máximo. Para obtener más información, consulte el apartado Netstat in the z/OS Communication Server IP System Administrator's Commands.  
 Las estadísticas sobre conexiones descartadas también están disponibles en los campos siguientes en Servicios TCP/IP: Estadísticas de recursos:
    - Conexiones descartadas (SOR\_CONNS\_DROPPED)
    - Hora de la última conexión descartada (SOR\_CONN\_LAST\_DROPPED)
  4. Si el valor ConnectionsDropped es demasiado alto, considere aumentar el valor del atributo de pendientes de TCPIP SERVICE.

## Diagnóstico de errores de tiempo de ejecución del solicitante de servicio

Lea esta sección si está teniendo problemas a la hora de enviar solicitudes de servicio web desde la aplicación de solicitante de servicio, o si está recibiendo mensajes de error SOAP desde el proveedor de servicios web.

### Acerca de esta tarea

Los problemas que se producen pueden deberse a errores en los servicios web individuales o problemas a nivel de transporte.

### Procedimiento

- Si está utilizando el mandato **INVOKE SERVICE** en el programa de aplicación, se devuelve un código RESP y RESP2 cuando hay un problema.
  1. Busque el significado de los códigos RESP y RESP2 para el mandato INVOKE SERVICE para que le indique cuál puede ser el problema.
  2. Compruebe el registro del sistema CICS para ver si hay mensajes que puedan ayudarle a determinar la causa del problema.
- Si no puede enviar un mensaje de solicitud SOAP y la interconexión está devolviendo un contenedor DFHERROR, se ha producido un problema cuando la interconexión ha intentado procesar el mensaje SOAP.
  1. Examine el contenido del contenedor DFHERROR. Debería contener un mensaje de error y datos que describan el problema.

2. ¿Ha introducido algún manejador de mensajes o programas de proceso de cabeceras nuevo en la interconexión? Si lo ha hecho, intente eliminar el nuevo programa y vuelva a ejecutar el servicio web para ver si esto resuelve el problema. Si el manejador de mensajes está intentando realizar algún proceso utilizando un contenedor que no está presente en la interconexión, o si está intentando actualizar un contenedor que es de sólo lectura, la interconexión detiene el proceso y devuelve un error en el contenedor DFHERROR. Los programas de proceso de cabeceras sólo pueden actualizar un conjunto limitado de contenedores en la interconexión. Consulte The header processing program interface para obtener más detalles.
  3. Si la aplicación de solicitante de servicio web no está utilizando el mandato **INVOKE SERVICE** para enviar una solicitud de servicio web, compruebe que ha creado todos los contenedores de control necesarios y que son del tipo de datos correcto. En concreto, compruebe que el contenedor DFHREQUEST tiene un tipo de datos de CHAR en lugar de BIT.
  4. Si la aplicación de solicitante de servicio web está utilizando el mandato **INVOKE SERVICE** y se devuelve un valor RESP de INVREQ y un código RESP2 de 14, esto indica que ha habido un error de conversión de datos. Consulte el apartado “Diagnóstico de errores de conversión de datos” en la página 637.
  5. Compruebe que un manejador de mensajes personalizado no ha invalidado el XML del mensaje SOAP durante el proceso de interconexión. CICS no realiza validaciones en los mensajes de salida de la interconexión. Si la aplicación utiliza el mandato **INVOKE SERVICE**, CICS genera el XML y tiene un formato correcto cuando el mensaje SOAP se coloca en el contenedor DFHREQUEST. Sin embargo, si tiene manejadores de mensajes adicionales que cambian el contenido del mensaje SOAP, no se valida en la interconexión.
- Si puede enviar un mensaje SOAP, pero recibe un error de transporte o de tiempo de espera excedido, esto normalmente se devuelve como un error SOAP. Si el programa está utilizando el mandato **INVOKE SERVICE**, CICS devuelve un valor RESP de TIMEDOUT y un código RESP2 de 2 para un error de tiempo de espera excedido, y un valor RESP de INVREQ y código RESP2 de 17 para un error de transporte.
    1. Compruebe que el punto final de red está presente.
    2. Asegúrese de que el atributo RESPWAIT en el recurso PIPELINE está configurado correctamente para cumplir los requisitos de la aplicación. El atributo RESPWAIT define cuánto tiempo espera CICS una respuesta del proveedor de servicio web antes de volver a la aplicación. Si no se especifica un valor, CICS utiliza los valores predeterminados de 10 segundos para HTTP y 60 segundos para WebSphere MQ. Sin embargo, CICS también tiene un tiempo de espera en el asignador para cada transacción, y si es menor que el valor predeterminado del protocolo que se está utilizando, CICS utiliza el tiempo de espera del asignador en su lugar.
  - Si puede enviar un mensaje SOAP, pero recibe una respuesta de error SOAP de vuelta del proveedor de servicios web que no esperaba, mire el contenido del contenedor DFHWS-BODY para obtener detalles del error SOAP.
    1. Si ha enviado un sobre SOAP completo en DFHREQUEST utilizando la interfaz DFHPIRT, asegúrese de que el mensaje de salida no contiene cabeceras SOAP duplicadas. Esto se puede producir cuando la interconexión de solicitante utiliza un manejador de mensajes SOAP 1.1 o SOAP 1.2. Los manejadores de mensajes SOAP añaden cabeceras SOAP, aunque la aplicación de solicitante de servicio ya las haya especificado en el sobre SOAP. En este escenario, puede:

- Eliminar el manejador de mensajes SOAP 1.1 o SOAP 1.2 de la interconexión. Esto afectará a otras aplicaciones de solicitante de servicio que utilicen esta interconexión.
- Eliminar las cabeceras SOAP del sobre SOAP que la aplicación pone en DFHREQUEST. CICS añade las cabeceras SOAP necesarias. Si desea realizar un proceso adicional en las cabeceras, puede utilizar la interfaz de programa de proceso de cabeceras.
- Utilice un mandato **WEB SEND** en lugar de la aplicación y renuncie al soporte de servicios web.
- Si piensa que el problema está relacionado con el envío o recepción de mensajes MTOM, consulte “Diagnóstico de errores de MTOM/XOP”.

## Diagnóstico de errores de MTOM/XOP

Se pueden producir errores de MTOM/XOP durante el tiempo de ejecución, en interconexiones en modo de solicitante o de proveedor.

### Antes de empezar

Si tiene problemas para configurar una interconexión compatible con MTOM/XOP, consulte el apartado “Diagnóstico de errores de despliegue” en la página 629.

### Acerca de esta tarea

#### Procedimiento

- Si puede enviar un mensaje de solicitud de servicio web en formato MTOM, pero recibe un mensaje de error de SOAP del proveedor de servicios web, busque en el contenido del contenedor DFHWS-BODY para ver los detalles del error de SOAP.
  1. ¿Puede recibir el proveedor de servicios web mensajes de MIME Multipart/Related? Si el proveedor de servicios web no admite el formato MTOM, el error que reciba variará en función de la implementación. Si el proveedor de servicios web es otra aplicación CICS, el error de SOAP indicaría que el mensaje MIME no es un tipo de contenido válido.
  2. Si el proveedor de servicios web puede recibir mensajes MIME, compruebe si la interconexión está enviando el mensaje en modalidad directa o de compatibilidad. Si está enviando un mensaje MTOM en modalidad directa, podría haber problemas con el XML.
  3. Para averiguar si el problema se encuentra en el XML, active la validación del servicio web. Esto hará que el mensaje MTOM se procese en modalidad de compatibilidad a través de la interconexión. Dentro de este procesamiento, el manejador MTOM analizará el contenido del mensaje para optimizar los datos base64binary. Si hay algún error en el XML, CICS colocará el error en el contenedor DFHERROR y emitirá un error de transporte MTOM en la interconexión.
  4. Examine el contenido del contenedor DFHERROR para ver si indica el problema que se ha producido. Si no hay suficiente información como para que pueda diagnosticar la causa del problema, ejecute un rastreo de nivel 2 del dominio de PI.
  5. Busque el PI del punto de rastreo 0C16. Describe el problema encontrado con más detalles y debería ayudarle a arreglar el problema con el XML proporcionado por la aplicación de solicitante.
- Si faltan archivos adjuntos binarios esperados del mensaje MTOM saliente, podría indicar que los datos binarios se consideran demasiado pequeños como

para optimizarse como archivo adjunto binario. CICS crea solo archivos adjuntos binarios para los datos que son lo bastante grandes como para justificar la sobrecarga de procesamiento que produciría su optimización en la interconexión. Los datos binarios de menos de 1.500 bytes de tamaño no se han optimizado.

- Si no puede enviar un mensaje MTOM saliente en modalidad de compatibilidad y la interconexión devuelve un contenedor DFHERROR, se ha producido un problema mientras la interconexión intentaba procesar el mensaje MTOM.
  1. Examine el contenido del contenedor DFHERROR. Debería contener un mensaje de error y datos que describan el problema.
  2. Compruebe que el XML del mensaje MTOM saliente sea válido. CICS no realiza validaciones en los mensajes salientes de la interconexión.
- Si recibe un mensaje DFHPI1100E, se ha producido un problema con las cabeceras MIME de un mensaje MTOM recibido por CICS. El mensaje CICS contiene la clase general de error MIME que se ha producido. Para averiguar el problema exacto que se ha producido:
  1. Si tiene el rastreo auxiliar activo en su región de CICS, compruebe si hay entradas de rastreo de excepción.
  2. Busque el punto de rastreo PI 1305. Describe la naturaleza del error de la cabecera MIME, la ubicación del error en la cabecera y hasta 80 bytes de texto anterior y posterior al error, a fin de ayudarle a comprender el contexto en el que se produjo el error.

Por ejemplo, el siguiente extracto de rastreo indica que el parámetro de inicio de tipo de contenido MIME no es válido debido a que no está entre comillas, aunque incluía caracteres que no son válidos fuera de una serie entrecomillada.

PI 1305 PIMM \*EXC\* - MIME\_PARSE\_ERROR -

```
TASK-01151 KE_NUM-0214 TCB-QR /009C7B68 RET-9C42790A TIME-10:33:41.3667303015 INTERVAL-00.0000053281 =000599=
1-0000 C5A79785 83A38584 40978199 819485A3 859940A5 8193A485 40A39692 85954096 *Expected parameter value token o*
0020 994098A4 96A38584 40A2A399 899587 *r quoted string *
2-0000 D4C9D4C5 40A2A895 A381A740 85999996 994081A3 404EF0F0 F0F0F1F1 F2408995 *MIME syntax error at +0000112 in*
0020 40C39695 A38595A3 60A3A897 85408885 81848599 * Content-type header *
3-0000 5F626F75 6E646172 793B2074 7970653D 22617070 6C696361 74696F6E 2F786F70 * _boundary; type="application/xop*
0020 2B786D6C 223B2073 74617274 2D696E66 6F3D2261 70706C69 63617469 6F6E2F73 **xml"; start-info="application/s*
0040 6F61702B 786D6C22 3B207374 6172743D *oap+xml"; start=*
4-0000 3C736F61 70736C75 6E674074 6573742E 68757273 6C65792E 69626D2E 636F6D3E *<soapslung@test.hursley.ibm.com*>
0020 3B206368 61727365 743D7574 662D38 *; charset=utf-8 *
```

- El proceso de interconexión no es capaz de analizar un mensaje MTOM entrante y el solicitante de servicio web recibe un mensaje de error de SOAP. Esto indica que se ha producido un problema con el documento XOP en el mensaje MTOM. En modalidad directa, es el manejador de aplicación quien genera el error de SOAP. Si la interconexión se está ejecutando en modalidad de compatibilidad, es el manejador MTOM quien analiza el mensaje al crear el mensaje SOAP. En este caso, CICS emite un mensaje de error con el prefijo DFHPI y un error de SOAP.
  1. El mensaje de error con el prefijo DFHPI indica dónde está el error del documento XOP. Por ejemplo, podría tratarse de una cabecera MIME no válida o un archivo adjunto binario.
  2. Para averiguar la causa exacta del problema, compruebe si hay puntos de rastreo de excepción. En concreto, busque los puntos de rastreo que empiecen por PI 13xx. Esto describe la excepción que se ha producido con más detalles.

También puede ejecutar un rastreo de PI nivel 2 para establecer la secuencia de sucesos que condujeron al error, pero esto podría tener un impacto de rendimiento significativo y no se recomienda en regiones de producción.

## Diagnóstico de errores de conversión de datos

Los errores de conversión de datos se pueden producir durante el tiempo de ejecución al convertir un mensaje SOAP en un CICS COMMAREA o contenedor y desde un COMMAREA o contenedor en un mensaje SOAP.

### Antes de empezar

Los síntomas incluyen la generación de mensajes de error SOAP y mensajes CICS que indican que se ha producido un fallo.

### Acerca de esta tarea

Si tiene un problema de conversión de datos, realice los pasos siguientes:

### Procedimiento

1. Asegúrese de que el recurso WEBSERVICE está actualizado. Vuelva a generar el archivo de enlace de servicio web para el servicio web y vuelva a desplegarlo en CICS.
2. Asegúrese de que el servicio web remoto se ha generado utilizando la misma versión de documento de servicio web (WSDL) que el utilizado o generado por CICS.
3. Si está seguro de que el recurso WEBSERVICE está utilizando un archivo de enlace de servicio web actual:
  - a. Habilite la validación en tiempo de ejecución para el recurso WEBSERVICE utilizando el mandato SET WEBSERVICE(*name*) VALIDATION donde *name* es el nombre de recurso WEBSERVICE.
  - b. Compruebe los mensajes CICS DFHPI1001 o DFHPI1002 en el registro de mensajes. DFHPI1001 describe la naturaleza precisa del problema de conversión de datos y puede ayudarle a identificar el origen del error de conversión. DFHPI1002 indica que no se han encontrado problemas.
  - c. Cuando ya no necesita validación para el servicio web, utilice el mandato siguiente para desactivar la validación: SET WEBSERVICE(*name*) NOVALIDATION.
4. Si aún no ha determinado el motivo para el error de conversión, realice un seguimiento CICS del servicio web que falla. Busque las siguientes entradas de rastreo de excepción de dominio PI:  

```
PI 0F39 - PICC *EXC* - CONVERSION_ERROR
PI 0F08 - PIII *EXC* - CONVERSION_ERROR
```

Un error de conversión PICC indica que se ha producido un problema al transformar un mensaje SOAP de entrada en un COMMAREA o contenedor. Un error de conversión PIII indica que se ha producido un problema al generar un mensaje SOAP desde un COMMAREA o contenedor proporcionado por el programa de aplicación. En ambos casos, el punto de rastreo identifica el nombre del campo asociado al error de conversión y también puede identificar el valor que está provocando el problema. Si se produce alguno de estos puntos de rastreo, van seguidos de un error de conversión. Para obtener una posible interpretación de estos errores de conversión, consulte las explicaciones de los mensajes DFHPI1007 a DFHPI1010.

### Por qué se producen errores de conversión de datos

CICS valida los mensajes SOAP sólo en la medida que sean necesarios para confirmar que contienen XML bien formado y para transformarlos. Esto significa que es posible que un mensaje SOAP se valide correctamente utilizando el WSDL, pero después falla en el entorno de tiempo de ejecución y viceversa.

El recurso WEBSERVICE encapsula las instrucciones de correlación para permitir que CICS lleve a cabo la conversión de datos durante el tiempo de ejecución. Se produce un error de conversión cuando la entrada no coincide con los datos esperados, como se describe en el recurso WEBSERVICE.

Esta no coincidencia se puede producir por cualquiera de los siguientes motivos:

- Un mensaje SOAP recibido por CICS no está bien formado y no es válido cuando se comprueba en la descripción de servicio web (WSDL) asociada al recurso WEBSERVICE.
- Un mensaje SOAP recibido por CICS que está bien formado y es válido pero contiene valores que están fuera del rango del recurso WEBSERVICE.
- El contenido de un contenedor o COMMAREA no es coherente con el recurso WEBSERVICE y la estructura de lenguaje desde la que se ha generado el servicio web.

Por ejemplo, el documento WSDL puede especificar restricciones de rango en un campo, como un unsignedInt que sólo puede tener un valor entre 10 y 20. Si un mensaje SOAP contiene un valor de 25, la validación del mensaje SOAP puede provocar que se rechace como no válido. El valor 25 se acepta como valor válido para un entero y se pasa a la aplicación.

Un segundo ejemplo es donde el documento WSDL especifica una serie sin especificar una longitud máxima. DFHWS2LS supone una longitud máxima de 255 caracteres de forma predeterminada al generar el archivo de enlace de servicio web. Si el mensaje SOAP contiene 300 caracteres, aunque la comprobación en WSDL valide el mensaje ya que no se ha establecido una longitud máxima, se informará de un error al intentar transformar el mensaje ya que el valor no se ajusta al almacenamiento intermedio de caracteres asignado por CICS.

## Problemas de la página de códigos

CICS utiliza el valor del parámetro de inicialización del sistema **LOCALCCSID** para codificar los datos del programa de aplicación. Sin embargo, el archivo de enlace de servicio web está codificado en US EBCDIC (Cp037). Esto puede llevar a problemas con la conversión de datos cuando la página de códigos utilizada por el programa de aplicación codifica los caracteres de forma diferente a la página de códigos US EBCDIC. Para evitar este problema, puede utilizar el parámetro **CCSID** en los trabajos por lotes del asistente de servicios web para especificar una página de códigos diferente para codificar datos entre el programa de aplicación y el archivo de enlace de servicios web. El valor de este parámetro sustituye el parámetro de inicialización del sistema **LOCALCCSID** para ese recurso WEBSERVICE particular. El *valor* de **CCSID** debe ser un EBCDIC CCSID.

## Mensajes de error SOAP para errores de conversión

Si se produce un error de conversión durante el tiempo de ejecución y CICS está actuando como proveedor de servicios web, se emite un mensaje de error SOAP para el solicitante de servicio. Este mensaje de error SOAP incluye el mensaje que emite CICS.

El solicitante de servicio puede recibir uno de los siguientes mensajes de error SOAP:

- No se puede convertir un mensaje SOAP

Este mensaje de error implica que el mensaje SOAP no está bien formado y no es válido o sus valores están fuera de rango.

- No se pueden convertir datos salientes

Este mensaje de error implica que el contenido de un COMMAREA o contenedor no es coherente.

- La operación no es parte de un servicio web

Este mensaje de error es una variación especial de cuando CICS recibe un mensaje SOAP no válido.

Si CICS es el solicitante de servicio web, el mandato **INVOKE SERVICE** devuelve un código RESP de INVREQ y un valor RESP2 de 14.

---

## Resolución de problemas en servicios web JSON

Los problemas que pueda tener al implementar servicios web de JSON en CICS pueden producirse durante el proceso de despliegue o en el tiempo de ejecución cuando CICS está transformando los mensajes.

### Acerca de esta tarea

Esta información de determinación de problemas es específica de CICS. Si tiene que notificar un problema al servicio de soporte técnico de IBM, utilice la información incluida en los datos de MustGather para asegurarse de que ha recopilado toda la información de diagnóstico necesaria.

El soporte para CICS como proveedor de servicios para solicitudes de JSON se basa fundamentalmente en el soporte de CICS para servicios web de SOAP.

## Resolución de problemas de despliegue de JSON

Pueden producirse errores de despliegue si se intenta instalar un recurso PIPELINE o un recurso WEBSERVICE en CICS. Aquí se describen los errores de despliegue más frecuentes, incluido el síntoma del problema, la causa y la solución.

### Procedimiento

Se pueden producir los siguientes errores al instalar un recurso PIPELINE o WEBSERVICE en CICS:

- Recibe un mensaje de error DFHPI0914 al intentar instalar un recurso WEBSERVICE. El mensaje incluye algo de información sobre la causa del error de instalación.
  1. Compruebe que ha otorgado permiso a CICS para leer el archivo de enlace de servicio web en z/OS UNIX.
  2. Compruebe que el archivo de enlace de servicios web no esté dañado. Esto se puede producir, por ejemplo, si desea usar el FTP para transferir el archivo a z/OS UNIX en modo de texto en lugar de en modo binario.
  3. Compruebe que los archivos de enlace de servicio web con el mismo nombre no estén en diferentes directorios de recogida.
  4. Compruebe que no esté instalando un recurso de WEBSERVICE en modo de proveedor en una interconexión en modo de solicitante. Los archivos de enlace de servicio web especifican un valor **PROGRAM**, en tanto que los archivos en modo de solicitante no.
  5. Si va a utilizar DFHJS2LS o DFHLS2JS, compruebe los mensajes emitidos por el trabajo para ver si hay algún problema que sea necesario resolver antes de crear el recurso WEBSERVICE.

- El recurso PIPELINE no se instala y se recibe un mensaje de error DFHPI0700, DFHPI0712, DFHPI0714 o de otro tipo que indica que se ha encontrado un error en el recurso PIPELIN.
  1. Si ha recibido un mensaje de error DFHPI0700, deberá habilitar el soporte del lenguaje PL/I para su región de CICS. Este soporte es necesario para poder instalar recursos PIPELINE. Consulte Language Environment support for PL/I para obtener más información.
  2. Compruebe que ha otorgado autorización a CICS para acceder a los directorios de z/OS UNIX para leer los archivos de configuración de interconexión.
  3. Compruebe que el directorio que se especifica en el parámetro **wsdir** sea válido. En particular, compruebe que el directorio y los nombres de archivo en z/OS UNIX distingan entre mayúsculas y minúsculas.
  4. Compruebe que no tiene un recurso PIPELINE con el mismo nombre en un estado ENABLED en la región de CICS.
- El recurso PIPELINE se instala en un estado DISABLED. Se obtiene un mensaje de error del rango de DFHPI0702 a DFHPI0711.
  1. Compruebe que no haya errores en el archivo de configuración de interconexión. Los elementos del archivo de configuración de interconexión solo pueden aparecer en determinados lugares. Si los especifica de forma incorrecta, recibirá un mensaje de error DFHPI0702. Este mensaje incluye el nombre del elemento que está causando el problema. Consulte la descripción del elemento para asegurarse de que la ha codificado en el lugar correcto.
  2. Compruebe que no existan caracteres no imprimibles, como caracteres de horizontal, en el archivo de configuración de la interconexión.
  3. Asegúrese de que el XML sea válido. Si el XML no es válido, puede originar errores de análisis al intentar instalar el recurso PIPELINE.
  4. Asegúrese de que el archivo de configuración de interconexión esté codificado en US EBCDIC. Si intenta utilizar una codificación EBCDIC diferente, CICS no podrá procesar el archivo.

## Resolución de problemas con el asistente de JSON

Si experimenta problemas con el asistente de JSON, utilice las técnicas de resolución de problemas para diagnosticar el problema.

### Procedimiento

Los siguientes errores se puede producir al ejecutar el asistente de JSON:

- Recibe un código de retorno 0, 4, 8 o 12 al ejecutar los trabajos por lotes del asistente de JSON de CICS. Para obtener más información sobre el código de retorno, consulte “Códigos de retorno del asistente de JSON” en la página 643.
  1. Compruebe el registro de trabajos para ver los mensajes de error o de aviso. Busque las explicaciones detalladas de los mensajes. Las explicaciones por lo general describen las acciones que puede realizar para arreglar el problema.
  2. Asegúrese de que ha introducido valores correctos para cada uno de los parámetros del trabajo. Los valores de parámetro, como los nombres de archivo y los elementos de la descripción de servicio web, distinguen entre mayúsculas y minúsculas.
  3. Asegúrese de que se ha especificado una combinación de parámetros correcta.
- Recibe un código de retorno 1, 136 o 139 al ejecutar trabajos por lotes del asistente de JSON de CICS. Estos códigos de retorno indican que la JVM ha



fallado, normalmente porque no tiene suficiente almacenamiento disponible. El asistente de CICS requiere un tamaño de la región de JCL de al menos 300 MB.

1. Aumente el tamaño de la región o tenga en cuenta establecer el tamaño de la región en 0M.
  2. Compruebe si hay alguna salida IEFUSI activa, que puede limitar el tamaño de la región.
- Recibe un código de retorno 137 al ejecutar los trabajos por lotes del asistente de JSON de CICS. Este código de retorno significa que el trabajo ha excedido el tiempo de espera.
    1. Incremente el tiempo codificando el parámetro **TIME** de la sentencia EXEC del trabajo a **TIME=1440**, o incremente el valor de **MAXCPU**TIME en el miembro SYS1.PARMLIB(BPXPRMxx).
  - Recibe un mensaje en el rango DFHPI9700 - DFHPI9711 que indica un esquema JSON no válido o no compatible al ejecutar DFHJS2LS.
    1. Compruebe que el esquema JSON es válido. Podría hacerlo comprobándolo con la especificación de esquema JSON o utilizar una herramienta, por ejemplo, json-schema-validator.
    2. Compruebe que el esquema JSON es compatible con DFHJS2LS. Para obtener más información, consulte [datamapping.hll-json.cicsassistant](http://datamapping.hll-json.cicsassistant).

## Qué hacer a continuación

Una vez se haya resuelto el problema, vuelva a ejecutar el trabajo de proceso por lotes del asistente de JSON.

## Resolución de problemas relativos a las solicitudes de JSON

Si CICS rechaza JSON, como mensaje de entrada de un servicio web o cuando se utiliza con la interfaz enlazable para transformar JSON, es posible que el JSON no sea válido o no se ajuste al esquema.

### Acerca de esta tarea

Si CICS detecta un problema con el JSON suministrado por un solicitante de servicio web o un programa de aplicación, se devuelve un mensaje de error con detalles del problema. Cuando CICS actúa como proveedor de servicios web, se devuelve el mensaje de error a la aplicación cliente como una respuesta JSON. Para obtener más información, consulte el apartado “Respuestas de error devueltas al cliente” en la página 642. Cuando una aplicación llama a la interfaz enlazable para transformar JSON, se devuelve un código de error en el contenedor DFHJSON-ERROR y un mensaje detallado en el contenedor DFHJSON-ERRORMSG. Para obtener más información, consulte el apartado JSON transformer linkable interface. En ambos casos, los pasos para resolver el problema son los mismos y dependen del tipo de error.

### Procedimiento

- Compruebe que el JSON se ajusta a las restricciones impuestas por CICS. Para obtener más información, consulte el apartado JSON web service restrictions.
- Si se produce un error de análisis de JSON, compruebe que el JSON esté formado correctamente. El mensaje de error proporciona detalles sobre el problema; por ejemplo:  
"Expected a ',' or '}' at character 44 of {"inquireCatalogRequest":  
"myData } Puede utilizar una herramienta para validar la sintaxis JSON; por ejemplo, JSONLint.

- Si CICS detecta un error estructural al intentar correlación JSON con datos de aplicación, se emite un mensaje DFHPI1007 que ofrece detalles del error. Por ejemplo:

```
DFHPI1007 04/19/2013 15:14:42 IYK2ZKE1 00112 JSON to data transformation
failed because of incorrect input (UNDEFINED_ELEMENT Operation) for
WEBSERVICE SimpleMappings.
```

Compruebe que el JSON contenga los objetos y propiedades de JSON descritos por el esquema proporcionado en DFHJS2LS o generado por DFHLS2JS. Puede utilizar una herramienta para validar que el JSON se ajusta al esquema.

- Si no se produce ningún error, pero la aplicación recibe datos vacíos en algunos campo, compruebe que se haya proporcionado el correspondiente JSON. CICS no detectará la ausencia de las propiedades de JSON descritas en el esquema. Puede utilizar una herramienta de validación de JSON para comprobar esto como se describió anteriormente.
- Se ha detectado un error interno del servidor.  
Error interno del servidor.

```
CICS TS: Unhandled Pipeline Error
```

```
Release: 670
```

En el navegador, compruebe el mensaje DFHSJ1006. Compruebe que existe JVMServer y no está inhabilitado.

- En algunas circunstancias, CICS añade un elemento derivador a la solicitud o respuesta JSON durante el proceso interno. Este nunca será visible para la aplicación, pero en ocasiones sí en aparecerá en los mensajes de error. Por ejemplo:

```
"Error obtaining parser from data source:Expected a ':'
after a key at character 25 of {"DFHWrapper": { 12jb01c\":"...
```

En estas situaciones, debe omitirse el elemento DFHWrapper al determinar la causa del error.

## Qué hacer a continuación

Una vez corregido el JSON, vuelva a ejecutar la aplicación.

## Respuestas de error devueltas al cliente

Estas respuestas se devuelven si se producen errores durante el proceso del manejador JSON de CICS.

### Acerca de esta tarea

Si se producen errores durante el proceso del manejador JSON de CICS, CICS devuelve una respuesta al cliente que incluye información sobre el error. Se devuelve un código de estado HTTP de 500 (Error de servidor interno), y el cuerpo del mensaje incluye detalles del error en formato JSON, en función del tipo de error que se haya producido.

### Ejemplo

Si se produce un error antes o después de que CICS haya llamado a la interconexión Axis2, el mensaje incluirá información similar a la siguiente:

```
{
 "exception" : {
 "message" : "An exception has occurred while validating HTTP headers".
 "class" : "com.ibm.cicsts.axis2.Controller"
 }
}
```

Si se produce un error en algún otro momento del proceso, el mensaje incluirá información similar a un error de SOAP, con una sección de detalle que varía en función de la naturaleza del error. Esta puede contener un mensaje CICS como, por ejemplo:

```
{
 "Fault": {
 "faultstring": "Conversion from SOAP failed",
 "detail": {
 "CICSFault": "DFHPI1007 02/14/2013 17:51:47 IYK2ZKE1 00185 XML to data transformation
failed because of incorrect input (UNDEFINED_ELEMENT startItemRuff) for
WEBSERVICE json_inquireCatalogWrapper."
 }
 }
}
```

## Códigos de retorno del asistente de JSON

Si se produce un fallo al ejecutar los trabajos por lotes del asistente de JSON, se proporciona un código de retorno que indica el tipo de fallo. Esta información se incluye en el registro de trabajo.

### Códigos de retorno del programa asistente de JSON

En el caso de un error de despliegue, los recursos PIPELINE se instalan por lo general en un estado DISABLED y los recursos WEBSERVICE se instalan en un estado UNUSABLE. Los mensajes informativos y de error asociados con los trabajos por lotes del asistente de JSON de CICS están ubicados en el registro de trabajo. Los mensajes de error asociados con los recursos de instalación se encuentran en el registro del sistema.

El asistente de JSON emite los códigos de 0, 4, 8 o 12, los otros códigos los suelen emitir BPXBATCH, JVM o IEBGENER.

Los códigos emitidos por BPXBATCH entran en dos categorías principales; un código de menos de 128 indica un error de mandato, un código mayor o igual que 128 indica que el proceso ha finalizado con una señal. Para obtener más información sobre BPXBATCH y sus códigos de retorno, consulte z/OS UNIX System Services Command Reference.

Recibe un código de retorno 0, 4, 8 o 12 al ejecutar los trabajos por lotes del asistente de JSON.

| Código de retorno | Descripción                                                                                                                                                      |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0                 | El trabajo ha terminado correctamente.                                                                                                                           |
| 4                 | Aviso. El trabajo se ha completado correctamente, pero se han emitido uno o más mensajes de advertencia.                                                         |
| 8                 | Error de entrada. El trabajo no ha finalizado satisfactoriamente. Uno o más mensajes de error se han emitido durante la validación de los parámetros de entrada. |

| Código de retorno | Descripción                                                                                                             |
|-------------------|-------------------------------------------------------------------------------------------------------------------------|
| 12                | Error. El trabajo no ha finalizado satisfactoriamente. Uno o más mensajes de error se han emitido durante la ejecución. |

## Códigos de retorno de trabajo por lotes del asistente de JSON

Recibe un código de retorno 1, 136, 137 o 139 al ejecutar trabajos por lotes del asistente de JSON de CICS.

| Código de retorno | Descripción                                                                                                                                                                  |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1                 | La JVM ha fallado, por lo general, debido a que no hay suficiente almacenamiento disponible. El asistente de CICS requiere un tamaño de la región de JCL de al menos 300 MB. |
| 136               | La JVM ha fallado, por lo general, debido a que no hay suficiente almacenamiento disponible. El asistente de CICS requiere un tamaño de la región de JCL de al menos 300 MB. |
| 137               | El tiempo de espera del trabajo se ha superado.                                                                                                                              |
| 139               | La JVM ha fallado, por lo general, debido a que no hay suficiente almacenamiento disponible. El asistente de CICS requiere un tamaño de la región de JCL de al menos 300 MB. |

---

## Apéndice. Ejemplos de servicios web

CICS proporciona una serie de ejemplos que puede utilizar como punto de partida para desarrollar aplicaciones y configurar CICS.

Los ejemplos se categorizan de la siguiente manera:

### Aplicación de ejemplo de gestor de catálogos CICS

La aplicación de ejemplo de gestor de catálogos de CICS es una aplicación COBOL de trabajo diseñada para ilustrar las prácticas recomendadas al conectar aplicaciones CICS a clientes y servidores externos.

La aplicación base tiene una interfaz de usuario 3270, pero la estructura modular, con interfaces bien definidas entre los componentes, hace posible añadir más componentes. En concreto, la aplicación viene con un soporte de servicio web, diseñado para ilustrar cómo puede ampliar una aplicación existente en un entorno de servicios web.

- “Aplicación de ejemplo de gestor de catálogos CICS”

### Ejemplos de JSON

Utilice estos ejemplos para comprender las solicitudes JSON.

- “Ejemplos de JSON” en la página 690

---

### Aplicación de ejemplo de gestor de catálogos CICS

La aplicación de ejemplo de gestor de catálogos de CICS es una aplicación COBOL de trabajo diseñada para ilustrar las prácticas recomendadas al conectar aplicaciones CICS a clientes y servidores externos.

El ejemplo se construye alrededor de un simple catálogo de ventas y una aplicación de proceso de pedido, en el que un usuario puede realizar estas tareas:

- Listar los artículos de un catálogo.
- Preguntar por artículos individuales de un catálogo.
- Solicitar artículos del catálogo.

El catálogo se implementa como archivo VSAM.

La aplicación base tiene una interfaz de usuario 3270, pero la estructura modular, con interfaces bien definidas entre los componentes, hace posible añadir más componentes. En concreto, la aplicación viene con un soporte de servicio web, diseñado para ilustrar cómo puede ampliar una aplicación existente en un entorno de servicios web.

En este ejemplo, CICS Explorer se puede utilizar para instalar y desplegar la aplicación. CICS Explorer es una interfaz de conjunto de herramientas gráfica basada en Eclipse para CICS.

Opcional: los usuarios que no utilizan CICS Explorer pueden utilizar la transacción CEDA para corregir e instalar los recursos contenidos en los grupos DFH\$EXBS y DFH\$EXWS.

## La aplicación base

La aplicación base, con su interfaz de usuario 3270, suministra funciones con las que puede listar el contenido de un catálogo almacenado, seleccionar un artículo de la lista y especificar una cantidad para el pedido. La aplicación tiene un diseño modular, que facilita la ampliación de la aplicación para soportar una tecnología más nueva, como servicios web.

La Figura 30 muestra la estructura de la aplicación base.  
Los componentes de la aplicación base son:

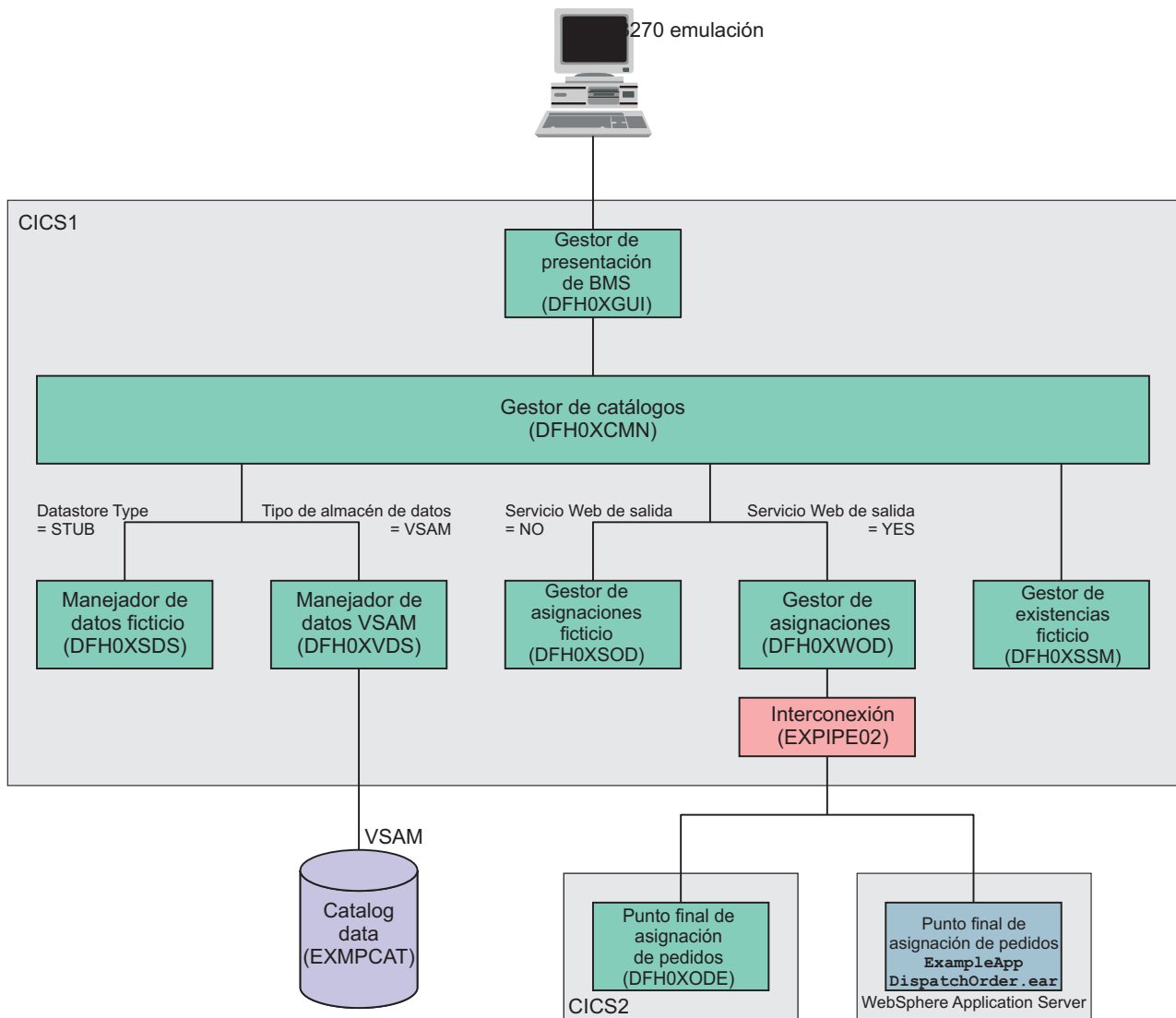


Figura 30. Estructura de la aplicación base

- Un gestor de presentación BMS (DFH0XGUI) que soporta un emulador o terminal 3270, y que interactúa con el programa de gestor de catálogos principal.
- Un programa de gestor de catálogos (DFH0XCMN) que es el núcleo de la aplicación de ejemplo y que interactúa con varios componentes de programa de fondo:
  - Un programa de manejador de datos que proporciona la interfaz entre el programa de gestor de catálogos y el almacén de datos. La aplicación base proporciona dos versiones de este programa. Son el programa de manejador

de datos VSAM (DFH0XVDS), que almacena datos en un conjunto de datos VSAM; y un manejador de datos ficticio (DFH0XSDS), que no almacena datos, pero devuelve respuestas válidas a su interlocutor. Las opciones de configuración le permiten elegir entre los dos programas.

- Un programa de gestor de asignación que proporciona una interfaz para asignar un pedido al cliente. De nuevo, las opciones de configuración le permite elegir entre las dos versiones de este programa: DFHX0WOD es un solicitante de servicio web que invoca un punto final de asignación de pedido remota, y DFHX0SOD es un programa ficticio que devuelve respuestas válidas a su interlocutor.

Hay dos puntos finales de asignación de pedidos equivalentes: DFH0XODE es un programa de proveedor de servicios CICS;

ExampleAppDispatchOrderV855.war es un archivo de archivado web Java que puede desplegarse en CICS Liberty JVM Server o entornos similares.

- Un programa de gestor de existencias ficticio (DFH0XSSM) que devuelve respuestas válidas a su interlocutor, pero que no realiza ninguna otra acción.

## **Gestor de presentación de BMS**

El gestor de presentación es responsable de todas las interacciones con el usuario a través de paneles 3270 BMS. En este programa no se toman decisiones de gestión.

El gestor de presentación BMS se puede utilizar de dos formas:

- Como parte de la aplicación base.
- Como un cliente de servicio web CICS que se comunica con la aplicación base utilizando mensajes SOAP.

## **Manejador de datos**

El manejador de datos proporciona la interfaz entre el gestor de catálogos y el almacén de datos.

La aplicación de ejemplo suministra dos versiones del manejador de datos:

- La primera versión utiliza un archivo VSAM como almacén de datos.
- La segunda versión es un programa ficticio que devuelve siempre los mismos datos sobre una consulta y no almacena los resultados de ninguna petición de actualización.

## **Gestor de asignaciones**

El gestor de asignaciones es responsable de asignar el pedido al cliente cuando se ha confirmado.

La aplicación de ejemplo suministra dos versiones del programa gestor de envío:

- La primera versión es un programa ficticio que devuelve una respuesta correcta a la persona que llama, pero no realiza ninguna otra acción.
- La segunda versión es un programa de solicitante de servicio que hace una solicitud a la dirección de punto final definida en el archivo de configuración.

## **Programa de asignación de pedidos**

El programa de asignación de pedidos es un programa de proveedor de servicios web responsable de asignar el elemento al cliente.

En la aplicación de ejemplo, el expedidor de pedidos es un programa ficticio que devuelve una respuesta correcta al llamante, pero no realiza ninguna otra acción. Hace que sea posible que todas las configuraciones de los servicios web de ejemplo sean operativas.

## Gestor de stocks

El gestor de stocks es responsable de gestionar la reposición del stock.

En el programa de ejemplo, el gestor de stocks es un programa ficticio que devuelve una respuesta correcta al llamante, pero no realiza ninguna otra acción.

## Configuración de aplicación

La aplicación de ejemplo incluye un programa que permite configurar la aplicación base.

## Instalación y configuración de la aplicación base

Antes de poder ejecutar la aplicación base, debe definir y llenar dos conjuntos de datos VSAM y crear dos recursos TRANSACTION.

## Creación y definición de conjuntos de datos VSAM

Se utilizan dos conjuntos de datos KSDS VSAM para definir y llenar la aplicación de ejemplo. Un conjunto de datos contiene información de configuración para la aplicación de ejemplo. El otro contiene el catálogo de ventas.

## Procedimiento

1. Localice el JCL para crear los conjuntos de datos VSAM. Durante la instalación de CICS, el JCL se coloca en la biblioteca *hlq.SDFHINST*:
  - El miembro DFH\$ECNF contiene el JCL para generar el conjunto de datos de configuración.
  - El miembro DFH\$ECAT contiene el JCL para generar el conjunto de datos de catálogo.
2. Modifique el JCL y los mandatos de servicios de método de acceso.
  - a. Suministre una tarjeta JOB válida.
  - b. Proporcione un calificador de alto nivel para los nombres del conjunto de datos en los mandatos de los servicios de método de acceso. Cuando se proporciona, el JCL utiliza un calificador de alto nivel de HLQ.

El mandato siguiente define el archivo de configuración:

```
DEFINE CLUSTER (NAME(hlq.EXMPLAPP.EXMPCONF) -
 TRK(1 1) -
 KEYS(9 0) -
 RECORDSIZE(350,350) -
 SHAREOPTIONS(2 3) -
 INDEXED -
) -
 DATA (NAME(hlq.EXMPLAPP.EXMPCONF.DATA) -
) -
 INDEX (NAME(hlq.EXMPLAPP.EXMPCONF.INDEX) -
)
```

donde *hlq* es el calificador de alto nivel de su elección.

El mandato siguiente define el archivo de catálogo:

```
DEFINE CLUSTER (NAME(hlq.EXMPLAPP.catname)-
 TRK(1 1) -
 KEYS(4 0) -
 RECORDSIZE(80,80) -
 SHAREOPTIONS(2 3) -
 INDEXED -
) -
 DATA (NAME(hlq.EXMPLAPP.catname.DATA) -
) -
 INDEX (NAME(hlq.EXMPLAPP.catname.INDEX) -
)
```



donde:

- *hlq* es el calificador de alto nivel de su elección
  - *nombre cat* es un nombre elegido por el usuario. El nombre utilizado en la aplicación de ejemplo según se suministra es EXMPCAT.
3. Ejecute ambos trabajos para crear y llenar los conjuntos de datos.
  4. Utilice CICS Explorer para crear una definición FILE para el archivo de catálogo.
    - a. Seleccione **Definiciones > Definiciones de archivo**. Pulse el botón derecho del ratón en la columna **Nombre** y pulse **Nuevo** para crear una nueva definición de archivo. Escriba EXAMPLE en el cuadro de texto **Grupo de recursos** y escriba EXMPCAT en el cuadro de texto **Nombre**. Pulse **Finalizar** para definir la definición FILE. Como alternativa, puede copiar la definición file del grupo DFH\$EXWS suministrado por CICS.
    - b. Efectúe una doble pulsación en el archivo EXMPCAT nuevo. En el editor Aplicación de ejemplo CICS de la definición de archivo (EXMPCAT), seleccione la pestaña **VSAM**. Escriba *hlq*.EXMPLAPP.EXMPCAT en el cuadro de texto **Nombre del conjunto de datos que se va a utilizar**.
    - c. Seleccione la pestaña **Atributos** y establezca las operaciones de los siguientes atributos en **Sí**:
      - Añadir
      - Examinar
      - Suprimir
      - Leer
      - Actualizar
    - d. Utilice los valores predeterminados para todos los demás atributos.
  5. Utilice el CICS Explorer para crear una definición FILE para el archivo de configuración.
    - a. Seleccione **Definiciones > Definiciones de archivo**. Pulse el botón derecho del ratón en la columna **Nombre** y pulse **Nuevo** para crear una nueva definición de archivo. Escriba EXAMPLE en el cuadro de texto **Grupo de recursos** y escriba EXMPCONF en el cuadro de texto **Nombre**. Pulse **Finalizar** para definir la definición FILE. Como alternativa, puede copiar la definición file del grupo DFH\$EXWS suministrado por CICS.
    - b. Efectúe una doble pulsación en el archivo EXMPCONF nuevo. En la ventana Aplicación de ejemplo CICS de la definición de archivo (EXMPCONF), seleccione la pestaña **VSAM**. Escriba *hlq*.EXMPLAPP.EXMPCONF en el cuadro de texto **Nombre del conjunto de datos que se va a utilizar**.
    - c. Seleccione la pestaña **Atributos** y establezca las operaciones de los siguientes atributos en **Sí**:
      - Añadir
      - Examinar
      - Suprimir
      - Leer
      - Actualizar
    - d. Utilice los valores predeterminados para todos los demás atributos.

## Resultados

Los conjuntos de datos se llenan y se crean las definiciones FILE para el archivo de catálogo y el archivo de configuración y están preparadas para su instalación.

Opcional: los usuarios que no utilizan CICS Explorer pueden utilizar la transacción CEDA para corregir e instalar los recursos contenidos en los grupos DFH\$EXBS y DFH\$EXWS.

## Definición de la interfaz 3270

La aplicación de ejemplo se suministra con una interfaz de usuario 3270 para ejecutar la aplicación y personalizarla. La interfaz de usuario consta de dos transacciones, EGUI y ECFG. Se utiliza una tercera transacción, ECLI, para el cliente de servicio web de CICS.

## Acerca de esta tarea

Opcional: los usuarios que no utilizan CICS Explorer pueden utilizar la transacción CEDA para corregir e instalar los recursos contenidos en los grupos DFH\$EXBS y DFH\$EXWS.

## Procedimiento

1. Cree definiciones TRANSACTION para las siguientes transacciones utilizando CICS Explorer. El funcionamiento correcto de la aplicación de ejemplo no depende de los nombres de las transacciones, así que puede utilizar nombres diferentes.

- a. Copie las definiciones para la transacción EGUI del grupo proporcionado por CICS, DFH\$EXBS, pulsando el botón derecho del ratón en DFH\$EXBS en la vista Definiciones del grupo de recursos. Seleccione **Nuevo > Definición de transacción**. Escriba EGUI en el cuadro de texto **Nombre** y DFH0XGUI en el cuadro de texto **Nombre de programa**. Pulse **Finalizar** para crear la definición EGUI TRANSACTION.
- b. Copie las definiciones para la transacción ECFG del grupo proporcionado por CICS, DFH\$EXBS, pulsando el botón derecho del ratón en DFH\$EXBS en la ventana Definiciones del grupo de recursos. Seleccione **Nuevo > Definición de transacción**. Escriba ECFG en el cuadro de texto **Nombre** y DFH0XCFG en el cuadro de texto **Nombre de programa**. Pulse **Finalizar** para crear la definición ECFG TRANSACTION.
- c. Opcional: Copie las definiciones para transacciones ECLI desde el grupo proporcionado por CICS, DFH\$EXWS, pulsando el botón derecho del ratón en DFH\$EXWS en la vista Definiciones del grupo de recursos. Seleccione **Nuevo > Definición de transacción**. Escriba ECLI en el cuadro de texto **Nombre** y DFH0XCUI en el cuadro de texto **Nombre de programa**. Pulse **Finalizar** para crear la definición de transacción ECLI.

Utilice los valores predeterminados para todos los demás atributos.

2. Opcional: Si no desea utilizar el programa de autoinstalación, copie las definiciones PROGRAM para los programas de aplicación base y las definiciones MAPSET para las correlaciones BMS desde el grupo proporcionado por CICS, DFH\$EXBS.
  - a. Copie las definiciones de recurso MAPSET para las correlaciones BMS en los miembros DFH0XS1, DFH0XS2 y DFH0XS3. Para obtener detalles de lo que hay en cada miembro, consulte “Componentes de la aplicación base” en la página 680.
  - b. Copie las definiciones de recurso PROGRAM para los siguientes programas COBOL.

Tabla 16. Miembros de SDFHSAMP que contienen el origen COBOL para la aplicación base

| Nombre de miembro | Descripción                                                                                                                                                                                                                                                                    |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DFH0XCFG          | Programa invocado por la transacción ECFG para leer y actualizar el archivo de configuración VSAM.                                                                                                                                                                             |
| DFH0XCMN          | Programa de controlador para la aplicación de catálogo. Todas las solicitudes pasan a través del programa de controlador.                                                                                                                                                      |
| DFH0XGUI          | Programa invocado por la transacción EGUI para gestionar el envío de correlaciones BMS al usuario de terminal y la recepción de correlaciones del usuario de terminal. Este programa enlaza con el programa DFH0XCMN.                                                          |
| DFH0XODE          | Una de las dos versiones del punto final para el servicio web de asignación de pedido. Esta es la versión que se ejecuta en CICS. Este programa establece el texto "Pedido en la asignación" en el COMMAREA de vuelta.                                                         |
| DFH0XSDS          | Un <i>apéndice</i> o versión ficticia del programa de almacén de datos que permite a la aplicación trabajar cuando el archivo de catálogo VSAM no se ha configurado. DFH0XSDS utiliza los datos definidos en el programa en lugar de los datos almacenados en un archivo VSAM. |
| DFH0XSOD          | Una versión de <i>apéndice</i> del programa de asignación de pedidos. Establece el código de retorno de COMMAREA en 0 y lo devuelve a su interlocutor. DFH0XSOD se utiliza cuando no se necesitan servicios web de salida.                                                     |
| DFH0XSSM          | Una versión de <i>apéndice</i> del programa de gestor de stock (reaprovisionamiento). DFH0XSSM establece el código de retorno en COMMAREA en 0 y lo devuelve a su interlocutor.                                                                                                |
| DFH0XVDS          | La versión VSAM del programa de almacén de datos. DFH0XVDS accede al archivo VSAM para llevar a cabo lecturas y actualizaciones del catálogo.                                                                                                                                  |
| DFH0XWOD          | La versión de servicio web del programa de asignación de pedidos. DFH0XWOD emite un EXEC CICS INVOKE WEBSERVICE para que el servicio web de salida invoque un asignador de pedidos.                                                                                            |

Utilice los valores predeterminados para todos los demás atributos.

- c. Opcional: Copie las definiciones PROGRAM para DFH0XCUI desde el grupo proporcionado por CICS, DFH\$EXWS. Utilice los valores predeterminados para todos los demás atributos. Este programa es necesario si desea utilizar la transacción ECLI que inicia el cliente de servicio web.

```
DIS G(DFH$EXWS)
ENTER COMMANDS
NAME TYPE GROUP
DFH0XCUI PROGRAM DFH$EXWS
ECLI TRANSACTION DFH$EXWS
EXMPPORT TCIPSERVICE DFH$EXWS
EXPIPE01 PIPELINE DFH$EXWS
EXPIPE02 PIPELINE DFH$EXWS
```

## Finalización de la instalación

Para completar la instalación, instale el grupo RDO que contiene las definiciones de recursos.

## Procedimiento

Pulse el botón derecho del grupo de recursos en la ventana Definiciones del grupo de recursos. Seleccione **Instalar**. Asegúrese de que el CICSplex es correcto y que selecciona la región de destino, a continuación, pulse **Aceptar**.

## Resultados

Su RDO está instalado y la aplicación está preparada para utilizarse.

Opcional: los usuarios que no utilizan CICS Explorer pueden utilizar la transacción CEDA para corregir e instalar los recursos contenidos en los grupos DFH\$EXBS y DFH\$EXWS.

## Configuración de la aplicación de ejemplo

La aplicación base incluye una transacción (ECFG) que puede utilizarse para configurar la aplicación de ejemplo.

## Antes de empezar

La transacción de configuración utiliza información en mayúsculas y minúsculas. Debe utilizar un terminal que pueda manejar información en mayúsculas y minúsculas correctamente.

## Acerca de esta tarea

Puede especificar varios aspectos de la aplicación de ejemplo. Por ejemplo:

- La configuración general de la aplicación, como el uso de servicios web
- Las direcciones de red utilizadas por los componentes de servicios web de la aplicación
- Los nombres de recursos, como por ejemplo el archivo utilizado para el almacén de datos
- Los nombres de programas utilizados para cada componente de la aplicación

Con la transacción de configuración, puede sustituir los componentes proporcionados por CICS de la aplicación de ejemplo por los suyos sin reiniciar la aplicación.

## Procedimiento

1. Especifique la transacción ECFG para iniciar la aplicación de configuración. CICS visualizará esta pantalla:

#### CONFIGURE CICS EXAMPLE CATALOG APPLICATION

```
Tipo de almacén de datos ==> VSAM STUB|VSAM
Servicio Web de salida ==> NO YES|NO
Gestor de catálogos ==> DFH0XCMN
Apéndice de almacén de datos ==> DFH0XSDS
VSAM de almacén de datos ==> DFH0XVDS
Apéndice de orden de asignación ==> DFH0XSOD
Servicio Web de orden de asignación ==>
DFH0XWOD
Gestor de Stocks ==> DFH0XSSM
Nombre de archivo VSAM ==> EXMPCAT
Dirección y puerto de servidor ==>
miservidor:99999
URI de servicio web de salida ==>
http://miservidor:80/exampleApp/dispatchOrder
==>
==>
==>
==>
==>
```

PF

3 END

12 CNCL

## 2. Rellene los campos.

### Tipo de almacén de datos

Especifique STUB si desea utilizar el programa de apéndice de almacén de datos.

Especifique VSAM si desea utilizar el programa de almacén de datos.

### Servicio Web de salida

Especifique YES si desea utilizar un servicio web remoto para la función de asignación de pedido; es decir, si desea que el programa de gestor de catálogos enlace con el programa de servicio web de asignación de pedido.

Especifique NO si desea utilizar un programa de apéndice para la función de asignación de pedido; es decir, si desea que el programa de gestor de catálogos enlace con el programa de apéndice de asignación de pedido.

### Gestor de catálogos

Especifique el nombre del programa de gestor de catálogos. El programa suministrado con la aplicación de ejemplo es DFH0XCMN.

### Apéndice de almacén de datos

Si ha especificado STUB en el campo **Tipo de almacén de datos**, especifique el nombre del programa de apéndice de almacén de datos. El programa suministrado con la aplicación de ejemplo es DFH0XSDS.

### VSAM de almacén de datos

Si ha especificado VSAM en el campo **Tipo de almacén de datos**, especifique el nombre del programa de almacén de datos VSAM. El programa suministrado con la aplicación de ejemplo es DFH0XVDS.

### Apéndice de envío de pedidos

Si ha especificado NO en el campo **Servicio web de salida**, especifique el nombre del programa de apéndice de asignación de pedido. El programa suministrado con la aplicación de ejemplo es DFH0XSOD.

### Servicio Web de envío de pedidos

Si ha especificado YES en el campo **Servicio Web de salida**, especifique el

nombre del programa que actúa como solicitante de servicio. El programa suministrado con la aplicación de ejemplo es DFH0XWOD.

#### Gestor de stocks

Especifique el nombre del programa de gestor de stock. El programa suministrado con la aplicación de ejemplo es DFH0XSSM.

#### Nombre de archivo VSAM

Si ha especificado VSAM en el campo **Tipo de almacén de datos**, especifique el nombre de la definición FILE de CICS. El nombre utilizado en la aplicación de ejemplo según se suministra es EXMPCAT.

#### Dirección y puerto de servidor

Si está utilizando el cliente de servicio web de CICS, especifique la dirección IP y el puerto del sistema en el que se despliega la aplicación de ejemplo como servicio web.

#### URI de servicio Web de salida

Si ha especificado YES en el campo **Servicio web de salida**, especifique la ubicación del servicio web que implementa la función de asignación de pedido. Si está utilizando el punto final de CICS proporcionado, establezca el **Servicio web de salida** en: `http://myserver:myport/exampleApp/dispatchOrder` donde *myserver* y *myport* son la dirección y puerto del servidor CICS.

## Ejecución de la aplicación de ejemplo con la interfaz BMS

La aplicación base se puede ejecutar utilizando la interfaz BMS.

### Procedimiento

1. Especifique la transacción EGUI desde un terminal CICS. Se visualiza el menú de aplicación de ejemplo:

APLICACIÓN DE CATÁLOGO DE EJEMPLO CICS - Menú principal

Seleccione una acción y pulse INTRO

Acción . . . . 1. Lista de artículos  
2. Número de artículo de pedido \_\_\_\_  
3. Salir

F3=SALIR F12=CANCELAR

Puede listar los elementos del catálogo, pedir un elemento o salir de la aplicación utilizando las opciones en el menú.

2. Escriba 1 y pulse Intro para seleccionar la opción Listar elementos. La aplicación visualizará una lista de ítems del catálogo.

APLICACIÓN DE CATÁLOGO DE EJEMPLO CICS - Consultar catálogo

Seleccione un solo artículo para el pedido con / y pulse INTRO

| Ítem<br>Pedido | Descripción                              | Coste  |   |
|----------------|------------------------------------------|--------|---|
| 0010           | Ball Pens Black 24pk                     | 2.90   | / |
| 0020           | Ball Pens Blue 24pk                      | 2.90   | — |
| 0030           | Ball Pens Red 24pk                       | 2.90   | — |
| 0040           | Ball Pens Green 24pk                     | 2.90   | — |
| 0050           | Pencil with eraser 12pk                  | 1.78   | — |
| 0060           | Highlighters Assorted 5pk                | 3.89   | — |
| 0070           | Laser Paper 28-lb 108 Bright 500/ream    | 7.44   | — |
| 0080           | Laser Paper 28-lb 108 Bright 2500/case   | 33.54  | — |
| 0090           | Blue Laser Paper 20lb 500/ream           | 5.35   | — |
| 0100           | Green Laser Paper 20lb 500/ream          | 5.35   | — |
| 0110           | IBM Network Printer 24 - Toner cart      | 169.56 | — |
| 0120           | Standard Diary: Week to view 8 1/4x5 3/4 | 25.99  | — |
| 0130           | Wall Planner: Eraseable 36x24            | 18.85  | — |
| 0140           | 70 Sheet Hard Back wire bound notepad    | 5.89   | — |
| 0150           | Sticky Notes 3x3 Assorted Colors 5pk     | 5.35   | — |

F3=SALIR F7=ATRÁS F8=ADELANTE F12=CANCELAR

3. Escriba / en la columna **Pedido** y pulse Intro para solicitar un elemento. La aplicación visualizará los detalles del artículo que debe solicitarse.

APLICACIÓN DE CATÁLOGO DE EJEMPLO CICS - Detalles del pedido

Especifique detalles del pedido y pulse INTRO

| Ítem<br>pedido | Descripción          | Coste | En       |
|----------------|----------------------|-------|----------|
| 0010           | Ball Pens Black 24pk | 2.90  | 0011 000 |

Cantidad de pedido: 5  
Nombre de usuario: CHRISB  
Dep facturación: CICSDEV1

F3=SALIR F12=CANCELAR

4. Si hay stock suficiente para realizar el pedido, entre la siguiente información:
  - a. Complete el campo **Cantidad pedida**. Especifique el número de artículos que desea solicitar.
  - b. Complete el campo **Nombre de usuario**. Entre una serie de 1 a 8 caracteres. La aplicación base no comprueba el valor especificado aquí.
  - c. Complete el campo **Departamento de facturación**. Entre una serie de 1 a 8 caracteres. La aplicación base no comprueba el valor especificado aquí.
5. Pulse Intro para enviar el pedido y volver al menú principal.
6. Pulse F3 para finalizar las aplicaciones.

## Soporte de servicios web para la aplicación de ejemplo

El soporte de servicio web amplía la aplicación de ejemplo, proporcionando dos versiones Java de un cliente inicial de servidor web, y una versión Java y COBOL del punto final del servicio web para el componente de asignador de pedido.

Las dos versiones del cliente web inicial y una versión del punto final de servicio web se proporcionan como archivos de archivado web (WARs) de Java que se ejecutan en el entorno de perfil web Java EE 6 proporcionado por la versión más reciente de WebSphere Application Server el servidor JVM más reciente de CICS TS Liberty. La segunda versión del punto final del servicio web se proporciona como un programa de aplicación de proveedor de servicios CICS (DFH0XODE).

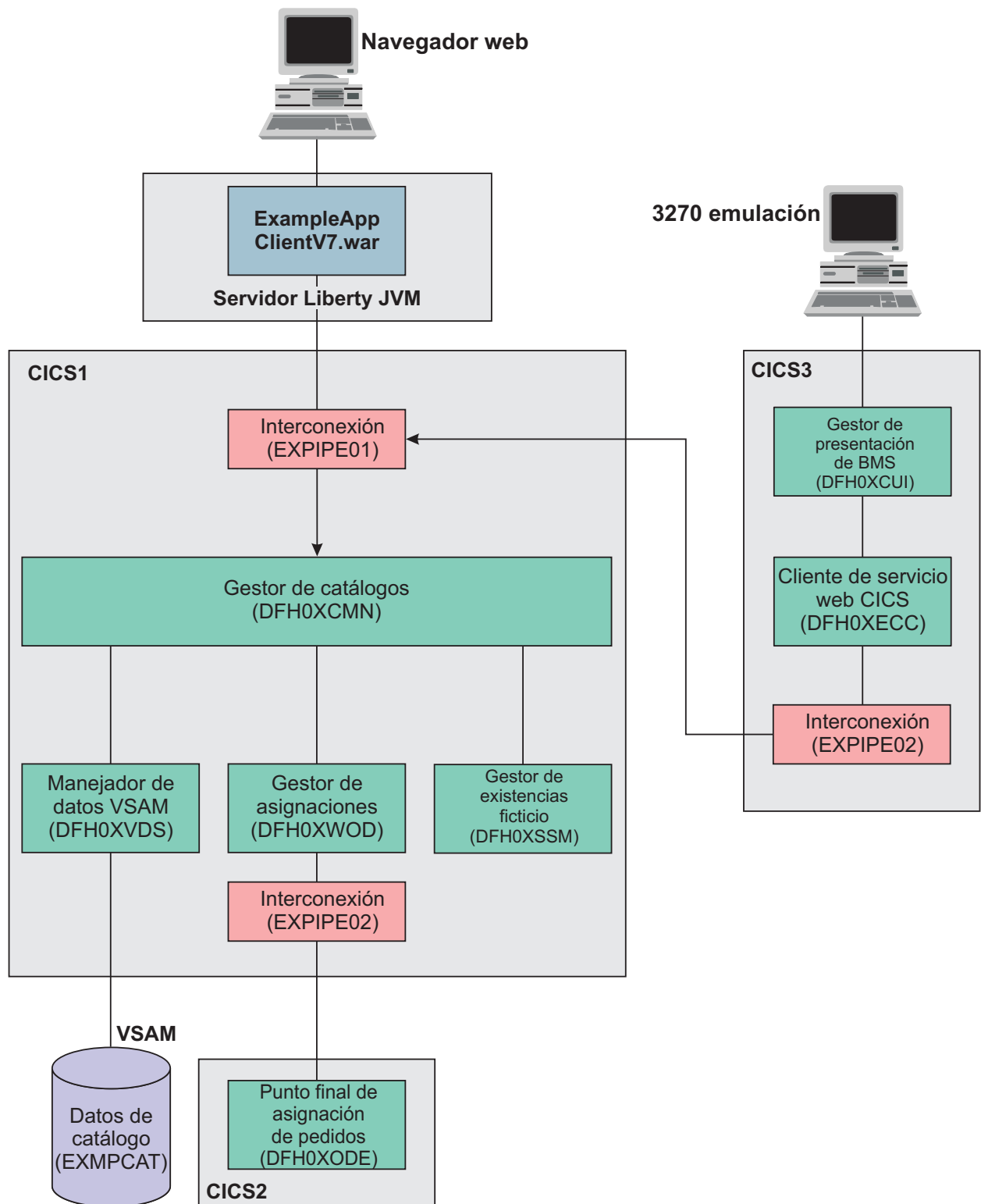
| Archivo                         | Descripción                                                             |
|---------------------------------|-------------------------------------------------------------------------|
| ExampleAppClientV855.war        | El cliente inicial de servicio web para el gestor de catálogos          |
| ExampleAppWrapperClientV855.war | El cliente inicial de servicio web para los derivadores de servicio web |
| ExampleAppDispatchOrderV855.war | Aplicación de proveedor de servicios web del asignador de pedidos       |

Estos WARs se exportan de proyectos web dinámicos. Para desplegar los archivos WAR, consulte .

Tenga en cuenta que necesitará habilitar la característica `jax-ws` en el servidor Liberty JVM añadiendo, por ejemplo `</feature> jaxws-2.2 </feature>` en el archivo de configuración del servidor Liberty, `server.xml`.

Figura 1 muestra una configuración de la aplicación de ejemplo con una versión del cliente web inicial y el proveedor de servicios CICS como punto final del servicio web del asignador de pedidos. También incluye un cliente de servicio web en un sistema CICS.



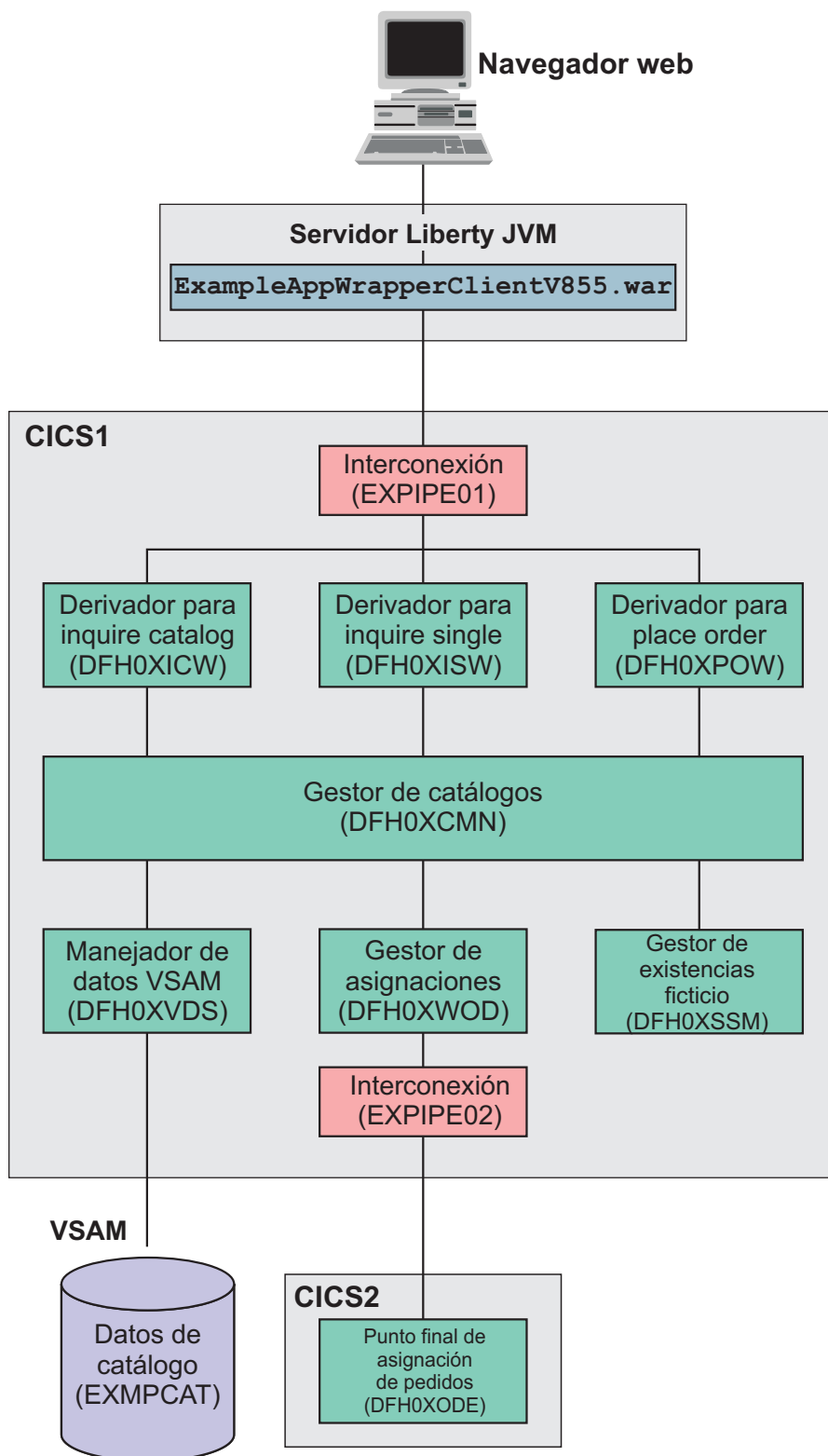


En esta configuración, se accede a la aplicación a través de dos clientes diferentes:

- Un cliente de navegador web conectado a un servidor Liberty JVM, en el que se despliega `ExampleAppClientV855.War`.

- Cliente de servicio web de CICS, DFH0XECC. Este cliente utiliza la misma lógica de presentación BMS que la aplicación base pero utiliza el módulo DFH0XCUI en lugar de DFH0XGUI.

Figura 2 muestra otra versión de cliente web inicial, con el proveedor de servicios CICS como punto final del servicio web del asignador de pedidos.



En esta configuración, el cliente de navegador está conectado al servidor Liberty JVM, en el que se despliega ExampleAppWrapperClientV855.war. En CICS, se

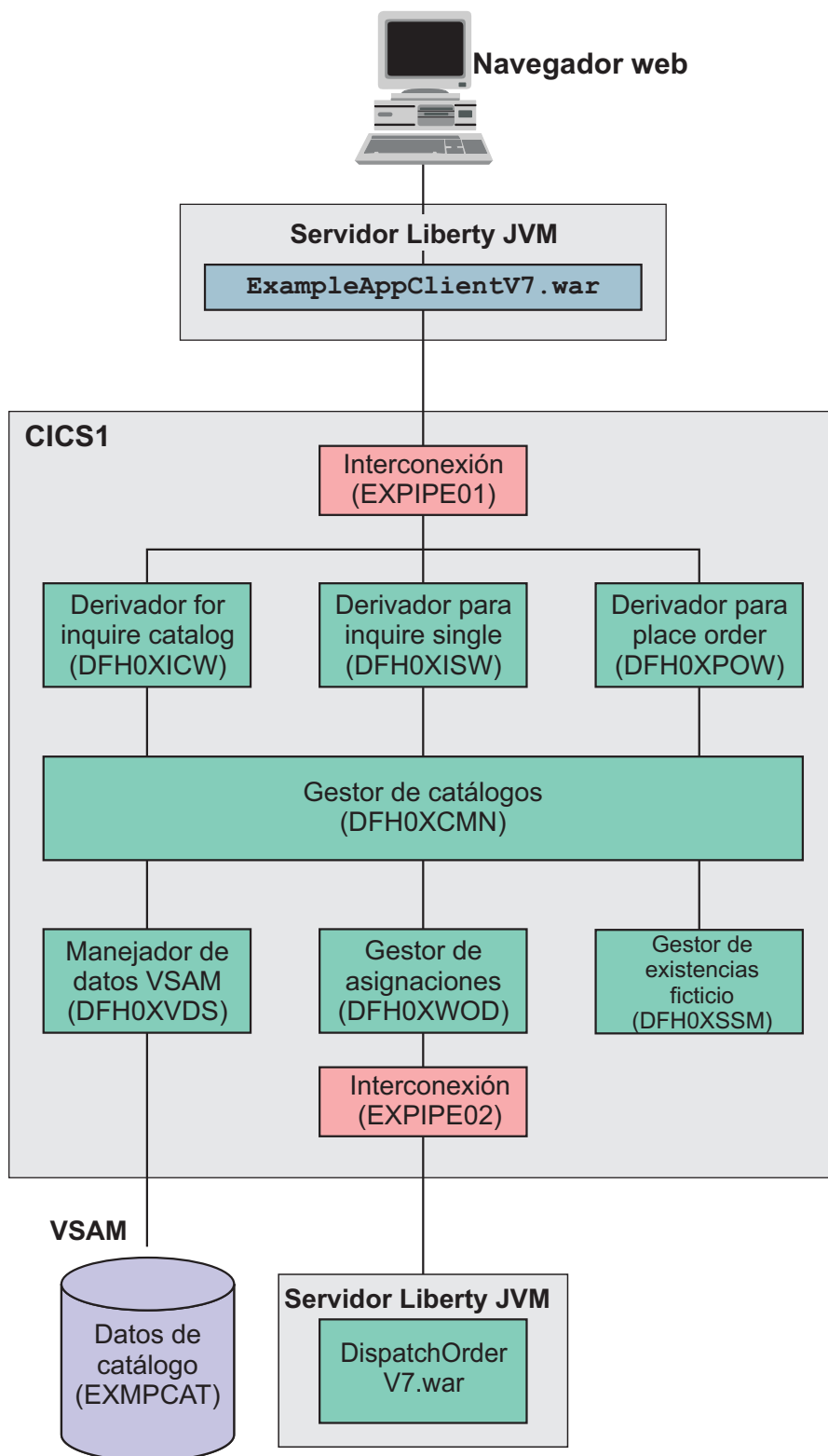
despliegan tres aplicaciones de derivador (para las funciones inquire catalog, inquire single y place order) como aplicaciones de proveedor de servicios. Que a su vez enlazan con la aplicación base.

Para que el gestor de aplicaciones en el sistema CICS invoque este punto final, debe cambiar la siguiente configuración utilizando la transacción de configuración ECFG:

- ¿Servicio Web de salida en YES?
- URI de servicio web de salida para el URI donde se está desplegando el punto final del Orden de asignación, por ejemplo, `http://cics2:8080/exampleApp/dispatchOrder`

Para obtener más detalles sobre la configuración de la aplicación de ejemplo, consulte *Configuring the example application*.

Figura 3 muestra una configuración de la aplicación de ejemplo con el cliente web inicial y el punto final del servicio web del asignador de pedidos en el servidor Liberty JVM.



En esta configuración, el cliente de navegador web está conectado al servidor Liberty JVM, en el que se despliega ExampleAppClientV855.war. El punto final del servicio web del asignador de pedidos ExampleAppDispatchOrderV855.war está instalado en el servidor Liberty JVM.

Para que el gestor de aplicaciones en el sistema CICS invoque este punto final, debe cambiar la siguiente configuración utilizando la transacción de configuración ECFG:

- ¿Servicio Web de salida en YES?
- URI de servicio web de salida para el URI donde se está desplegando el punto final del Orden de asignación, por ejemplo `http://mylibertyserver:9080/ExampleAppDispatchOrderV855/DispatchOrder`

Para obtener más detalles sobre la configuración de la aplicación de ejemplo, consulte *Configuring the example application*.

## Configuración del soporte de página de códigos

Tal como se suministra, la aplicación de ejemplo utiliza dos juegos de caracteres codificados. Debe configurar el sistema para habilitar la conversión de datos entre los dos juegos de caracteres.

### Acerca de esta tarea

Los juegos de caracteres codificados utilizados en la aplicación de ejemplo son:

- 037** EBCDIC Grupo 1: Estados Unidos, Canadá (z/OS), Holanda, Portugal, Brasil, Australia, Nueva Zelanda)
- 1208** UTF-8 Nivel 3

### Procedimiento

Añada las siguientes sentencias a la imagen de conversión del sistema z/OS:

```
CONVERSION 037,1208;
CONVERSION 1208,037;
```

Para obtener más información, consulte el apartado *Unicode data conversion by z/OS*.

## Definición de los programas derivadores y de cliente de servicio web

Si no está utilizando el programa de autoinstalación, debe definir las definiciones de recursos para los programas derivadores y de cliente de servicio web.

### Acerca de esta tarea

Opcional: los usuarios que no utilizan CICS Explorer pueden utilizar la transacción CEDA para corregir e instalar los recursos contenidos en los grupos DFH\$EXBS y DFH\$EXWS.

### Procedimiento

Use CICS Explorer para definir definiciones de recursos PROGRAM para programas derivadores, seleccionando **Definiciones > Definiciones de programa**. Pulse el botón derecho en la vista Definiciones de programa y seleccione **Nuevo** para crear una definición de programa nueva. Escriba un grupo CSD en el cuadro de texto **Grupo CSD**, y escriba el nombre de programa en el cuadro de texto **Nombre**. Pulse **Finalizar** para definir la definición PROGRAM. Cree definiciones para los siguientes programas COBOL:

Tabla 17. Miembros de SDFHSAMP que contienen el código de origen COBOL para módulos de derivador

| Nombre de miembro | Descripción                                        |
|-------------------|----------------------------------------------------|
| DFH0XECC          | Programa del cliente de servicios web              |
| DFH0XICW          | Programa derivado para el servicio inquireCatalog. |
| DFH0XISW          | Programa derivado para el servicio inquireSingle.  |
| DFH0XPOW          | Programa derivado para el servicio purchaseOrder.  |

## Instalación del soporte de servicio web

Antes de poder ejecutar el soporte de servicio web para la aplicación de ejemplo, debe crear dos directorios z/OS UNIX, y crear los recursos de CICS necesarios.

### Directorios z/OS UNIX:

El soporte de servicio web para la aplicación de ejemplo requiere un directorio de estantería y un directorio de recogida en z/OS UNIX.

El directorio de estantería se utiliza para almacenar los archivos de enlace WEBSERVICE asociados a recursos WEBSERVICE. Cada recurso WEBSERVICE está, a su vez, asociado con un PIPELINE. El recurso PIPELINE gestiona el directorio de estantería, cuyo contenido no debe cambiarse directamente. Varias PIPELINE pueden utilizar el mismo directorio de estantería, ya que CICS garantiza una estructura de directorio exclusiva debajo del directorio de estantería para cada PIPELINE.

El directorio de recogida es el directorio que contiene los archivos de enlace WEBSERVICE asociados a PIPELINE. Cuando se instala un PIPELINE, o en respuesta a un mandato **PERFORM PIPELINE SCAN**, la información de los archivos de enlace se utiliza para crear dinámicamente las definiciones WEBSERVICE y URIMAP asociadas con el PIPELINE.

La aplicación de ejemplo utiliza /var/cicsts para el directorio de estantería.

### Creación de la definición de interconexión:

La definición completa de una interconexión consta de un recurso PIPELINE y un archivo de configuración PIPELINE. El archivo contiene los detalles de los manejadores de mensajes que actúan en las solicitudes y respuestas de servicio web cuando pasan a través de la interconexión.

### Acerca de esta tarea

Opcional: los usuarios que no utilizan CICS Explorer pueden utilizar la transacción CEDA para corregir e instalar los recursos contenidos en los grupos DFH\$EXBS y DFH\$EXWS.

La aplicación de ejemplo utiliza el manejador SOAP 1.1 proporcionado para tratar con sobres SOAP de solicitudes de entrada y salida. CICS proporciona archivos de configuración de interconexión de ejemplo, que puede utilizar en el proveedor de servicios y en el solicitante de servicio.

Varios servicios web pueden compartir una única interconexión, por lo tanto, sólo necesita definir una interconexión para las solicitudes de entrada de la aplicación de ejemplo. Sin embargo, debe definir una segunda interconexión para las

solicitudes de salida porque no se puede configurar una única interconexión para que sea una interconexión de proveedor y de solicitante a la vez.

Si desea utilizar interconexiones basadas en Java, debe especificar el archivo de configuración de proveedor de servicios de ejemplo `basicsoap11javaprovider.xml` en lugar de `basicsoap11provider.xml` en el paso 1b. Y especifique el archivo de configuración del solicitante de servicio de ejemplo `basicsoap11javarequester.xml` en lugar de `basicsoap11requester.xml` en el paso 2b. Para obtener más información sobre los archivos de configuración de ejemplo, consulte Pipeline configuration files. Además, si desea utilizar el manejador de aplicación Axis2 en la interconexión basada en Java, debe sustituir EXPIPE01 por EXPIPE03 en el paso 1a y EXPIPE02 por EXPIPE04 en el paso 2a

### Procedimiento

1. Utilice CICS Explorer para crear una definición de interconexión para el proveedor de servicios.
  - a. Cree una definición PIPELINE para los programas derivadores utilizando CICS Explorer seleccionando **Definiciones > Definiciones de interconexión**. Pulse el botón derecho en la vista Definiciones de interconexión y seleccione **Nuevo** para crear una definición de interconexión nueva. Escriba DFH\$EXWS en el cuadro de texto **Grupo de recursos**, y escriba EXPIPE01 en el cuadro de texto **Nombre**. Pulse **Finalizar** para crear la definición PIPELINE. Como alternativa, puede copiar la definición PIPELINE del grupo DFH\$EXWS suministrado por CICS. Pulse el botón derecho en DFH\$EXWS en la vista Definición de grupo de recursos y seleccione **Nuevo > Definición de interconexión**.
  - b. Efectúe doble pulsación en la definición PIPELINE y seleccione la pestaña **Atributos** desde el editor Definición de interconexión (EXPIPE01). En **Detalles**, el **Archivo de configuración** debe establecerse en la ubicación de los archivos de ejemplo `/usr/lpp/cicsts/samples/pipelines/basicsoap11provider.xml`, donde `/usr/lpp/cicsts` es la ruta a los archivos del directorio, **Estantería** debe ser `/var/cicsts/`, **Estado** debe ser **ENABLED**, y **Directorio WS** debe ser `/usr/lpp/cicsts/samples/webservices/wsbind/provider/`.

Las entradas de z/OS UNIX distinguen entre mayúsculas y minúsculas y asumen una raíz de instalación de CICS z/OS UNIX predeterminada de `/usr/lpp/cicsts`.
2. Utilice CICS Explorer para crear una definición PIPELINE para el solicitante de servicio.
  - a. Cree una definición PIPELINE para los programas derivadores utilizando CICS Explorer seleccionando **Definiciones > Definiciones de interconexión**. Pulse el botón derecho en la vista Definiciones de interconexión y seleccione **Nuevo** para crear una definición de interconexión nueva. Escriba DFH\$EXWS en el cuadro de texto **Grupo de recursos**, y escriba EXPIPE02 en el cuadro de texto **Nombre**. Pulse **Finalizar** para crear la definición PIPELINE. Como alternativa, puede copiar la definición PIPELINE del grupo DFH\$EXWS suministrado por CICS.
  - b. Efectúe doble pulsación en la definición PIPELINE y seleccione la pestaña **Atributos** desde el editor Definición de interconexión (EXPIPE02). En **Detalles**, el **Archivo de configuración** debe establecerse en la ubicación de los archivos de ejemplo `/usr/lpp/cicsts/samples/pipelines/basicsoap11requester.xml`, donde `/usr/lpp/cicsts` es la ruta para los



archivos del directorio, **Estantería** debe ser /var/cicsts/, **Estado** debe ser ENABLED, y **Directorio WS** debe ser /usr/lpp/cicsts/samples/webservices/wsbind/requester/.

### Creación de un servicio TCP/IP:

Puesto que el cliente conecta su servicio web sobre un transporte HTTP, debe definir un servicio TCP/IP para recibir el tráfico HTTP de entrada.

### Procedimiento

Utilice CICS Explorer para crear una definición TCPIPSERVICE para manejar solicitudes HTTP de entrada.

1. Cree una definición TCPIPSERVICE seleccionando **Definiciones > Definiciones de servicio TCP/IP**. Pulse el botón derecho en la vista Definiciones de servicio TCP/IP y seleccione **Nuevo** para crear una definición nueva. Escriba DFH\$EXWS en el cuadro de texto **Grupo de recursos**, y escriba EXMPPORT en el cuadro de texto **Nombre**. Debe especificar un número de puerto; escriba el número de cualquier puerto no utilizado en el sistema CICS. Pulse **Finalizar** para crear la definición TCPIPSERVICE.
2. Efectúe una doble pulsación en la definición TCPIPSERVICE. En la pestaña **Atributos** en el editor Definición de servicio TCP/IP (EXMPPORT), establezca los siguientes atributos:
  - Urm** debe ser DFHWBAAX
  - Protocolo** debe ser HTTP
  - Transacción** debe ser CWXN
3. Utilice los valores predeterminados para todos los demás atributos.

Opcional: los usuarios que no utilizan CICS Explorer pueden utilizar la transacción CEDA para corregir e instalar los recursos contenidos en los grupos DFH\$EXBS y DFH\$EXWS.

### Instalación dinámica de los recursos WEBSERVICE y URIMAP:

Cada función que se expone como servicio web requiere un recurso WEBSERVICE para correlacionar entre el XML de entrada de SOAP BODY y la interfaz COMMAREA del programa, y un recurso URIMAP que direcciona las solicitudes entrantes al servicio web e interconexión correctos. Aunque puede utilizar la definición de recursos en línea (RDO) para definir e instalar los recursos WEBSERVICE y URIMAP, también puede hacer que CICS los cree dinámicamente cuando instala un recurso de interconexión.

### Acerca de esta tarea

Opcional: los usuarios que no utilizan CICS Explorer pueden utilizar la transacción CEDA para corregir e instalar los recursos contenidos en los grupos DFH\$EXBS y DFH\$EXWS.

### Procedimiento

1. Utilice CICS Explorer para instalar los recursos PIPELINE.
  - a. Seleccione **Definiciones > Definiciones de interconexión**. Pulse el botón derecho del ratón en la definición EXPIPE01 PIPELINE en la vista Definiciones de interconexión y seleccione **Instalar**. Seleccione la región de CICS de destino seleccionando la casilla de verificación. Pulse **Aceptar** para instalar PIPELINE.

**Nota:** Si ha creado definiciones de interconexión basadas en Java en “Creación de la definición de interconexión” en la página 663, a continuación, pulse el botón derecho en la definición EXPIPE03 PIPELINE en la vista Definiciones de interconexión.

- b. Repita este proceso para la definición EXPIPE02 PIPELINE o EXPIPE04 para las interconexiones basadas en Java.

Al instalar cada recurso PIPELINE, CICS explora el directorio especificado en el atributo PIPELINE WSDIR (el directorio de recogida). Para cada archivo de enlace WEBSERVICE en el directorio, es decir, para cada archivo con el sufijo .wsbind, CICS instala un recurso WEBSERVICE y un recurso URIMAP si estos recursos no existen.

El recurso URIMAP proporciona a CICS la información necesaria para asociar el recurso WEBSERVICE con un URI concreto. Los recursos existentes se sustituyen si la información del archivo de enlace es más reciente que los recursos existentes.

Se instala un segundo recurso URIMAP opcional si se ha copiado un archivo WSDL o un archivo de archivado WSDL en el directorio de recogida. Este recurso URIMAP proporciona a CICS la información para asociar el archivo de archivado WSDL o el documento WSDL con un URI específico para que los solicitantes externos puedan utilizar el URI para descubrir el archivo de archivado WSDL o el documento WSDL.

Cuando PIPELINE se inhabilita o descarta más tarde, también se descartan todos los recursos WEBSERVICE y URIMAP asociados.

2. Si ya ha instalado el recurso PIPELINE, utilice el mandato **PERFORM PIPELINE SCAN** para iniciar la exploración del directorio de recogida PIPELINE.

Cuando haya instalado los recursos PIPELINE, se instalan en el sistema los siguientes recursos WEBSERVICE y los recursos asociados URIMAP para la interconexión de proveedor:

```
dispatchOrder
dispatchOrderEndpoint
inquireCatalog
inquireCatalogClient
inquireCatalogWrapper
inquireSingle
inquireSingleClient
inquireSingleWrapper
placeOrder
placeOrderClient
placeOrderWrapper
```

Los nombres de los recursos WEBSERVICE se derivan de los nombres de los archivos de enlace WEBSERVICE; los nombres de los recursos URIMAP se generan dinámicamente. Se genera un URIMAP adicional para cada documento WSDL que existe en el directorio de recogida de la interconexión. Puede ver los recursos seleccionando **Operaciones > Servicios web** para abrir la vista Servicios web. Pulse el botón derecho en el recurso WEBSERVICE y seleccione **Abrir relacionados > Correlación de URI**.

La vista CICS Explorer muestra los nombres del recurso PIPELINE, el recurso URIMAP y el programa de destino asociado a cada servicio web. En este ejemplo, no hay URIMAP o programa de destino para WEBSERVICE(dispatchOrder) porque el recurso WEBSERVICE es para una solicitud de salida.

WEBSERVICE(dispatchOrderEndpoint) representa la implementación CICS local del servicio del orden de asignación.

### Creación de recursos WEBSERVICE con una definición de recursos en línea (RDO):

Como alternativa al uso del mecanismo de exploración de interconexión para instalar recursos WEBSERVICE, puede crear e instalarlos utilizando la definición de recursos en línea (RDO).

#### Antes de empezar

**Importante:** Si utiliza RDO para definir los recursos WEBSERVICE y URIMAP, debe asegurarse de que sus archivos de enlace de servicio web **no** están en el directorio de recogida de PIPELINE. Esto garantiza que los recursos WEBSERVICE y URIMAP no se instalaron dinámicamente durante una exploración de interconexión del directorio de recogida. De forma alternativa, puede garantizar que no se especifica ningún valor para WSDIR en PIPELINE. Sin embargo, si no especifica un valor para WSDIR, no se produce ninguna explotación de interconexión del directorio de recogida. Por lo tanto, todos los recursos WEBSERVICE y URIMAP deben crearse e instalarse utilizando RDO.

#### Procedimiento

1. Utilice CICS Explorer para crear una definición WEBSERVICE para la función inquire catalog de la aplicación de ejemplo.
  - a. Cree una definición WEBSERVICE utilizando CICS Explorer seleccionando **Definiciones > Definición servicio web**.
  - b. Pulse el botón derecho en la vista Definiciones de servicio web y seleccione **Nuevo** para crear una nueva definición WEBSERVICE.
  - c. Escriba DFH\$EXWS en el cuadro de texto **Grupos de recurso**, escriba EXINQWS en el cuadro de texto **Nombre** y escriba EXPIPE01 en el cuadro de texto **Interconexión** o escriba EXPIPE03 para interconexiones basadas en Java. Debe entrar el atributo WSBind antes de poder crear la definición WEBSERVICE. En el tipo de cuadro de texto **WSBind File**, escriba /usr/lpp/cicsts/samples/webservices/wsbind/provider/inquireCatalog.wsbind.
  - d. Pulse **Finalizar** para crear la definición WEBSERVICE.
2. Repita los pasos anteriores para cada una de las funciones siguientes de la aplicación de ejemplo.

| Función                                  | Nombre de WEBSERVICE | Atributo PIPELINE   | Atributo WSBind                                                                  |
|------------------------------------------|----------------------|---------------------|----------------------------------------------------------------------------------|
| INQUIRE SINGLE ITEM                      | EXINQWS              | EXPIPE01 o EXPIPE03 | /usr/lpp/cicsts/samples/webservices/wsbind/provider/inquireSingle.wsbind         |
| PLACE ORDER                              | EXORDRWS             | EXPIPE01 o EXPIPE03 | /usr/lpp/cicsts/samples/webservices/wsbind/provider/placeOrder.wsbind            |
| DISPATCH STOCK                           | EXODRQWS             | EXPIPE02 o EXPIPE04 | /usr/lpp/cicsts/samples/webservices/wsbind/requester/dispatchOrder.wsbind        |
| Punto final de DISPATCH STOCK (opcional) | EXODEPWS             | EXPIPE01 o EXPIPE03 | /usr/lpp/cicsts/samples/webservices/wsbind/provider/dispatchOrderEndpoint.wsbind |

## Creación de recursos URIMAP con la definición de recursos en línea (RDO):

Como alternativa al uso del mecanismo de exploración de interconexión para instalar recursos URIMAP, puede crear e instalarlos utilizando la definición de recursos en línea (RDO).

### Antes de empezar

**Importante:** Si utiliza RDO para definir los recursos WEBSERVICE y URIMAP, debe asegurarse de que sus archivos de enlace de servicio web **no** están en el directorio de recogida de PIPELINE. Esto garantiza que los recursos WEBSERVICE y URIMAP no se instalaron dinámicamente durante una exploración de interconexión del directorio de recogida. De forma alternativa, puede garantizar que no se especifica ningún valor para WSDIR en PIPELINE. Sin embargo, si no especifica un valor para WSDIR, no se produce ninguna explotación de interconexión del directorio de recogida. Por lo tanto, todos los recursos WEBSERVICE y URIMAP deben crearse e instalarse utilizando RDO.

### Procedimiento

1. Utilice CICS Explorer para crear una definición URIMAP para la función inquire catalog de la aplicación de ejemplo.
  - a. Cree una definición URIMAP en CICS Explorer seleccionando **Definiciones > Definición de correlación URI**.
  - b. Pulse el botón derecho en la vista Definiciones de correlación URI y seleccione **Nuevo** para crear una definición URIMAP nueva.
  - c. Escriba INQCURI en el cuadro de texto **Nombre** y \* en el cuadro de texto **Host**. Debe entrar el atributo **Vía de acceso** antes de poder crear la definición URIMAP. En el cuadro de texto **Vía de acceso**, escriba /exampleApp/inquireCatalog. **Uso** debe establecerse en **Interconexión**; el recurso PIPELINE es EXPIPE01 o EXPIPE03 para interconexiones basadas en Java.
  - d. Pulse **Finalizar** para finalizar la definición URIMAP.
  - e. Efectúe doble pulsación en el nuevo recurso URIMAP para abrir el Editor. En la pestaña **Atributos** del Editor, establezca el atributo **Servicio web** en EXINQCWS y **Servicio TCP/IP** en SOAPPORT.
2. Repita los pasos anteriores para cada una de las funciones restantes de la aplicación de ejemplo. Utilice los siguientes nombres para los recursos URIMAP:

| Función                                  | Nombre de URIMAP |
|------------------------------------------|------------------|
| INQUIRE SINGLE ITEM                      | INQSURI          |
| PLACE ORDER                              | ORDRURI          |
| DISPATCH STOCK                           | No necesario     |
| Punto final de DISPATCH STOCK (opcional) | ODEPURI          |

3. Especifique los siguientes atributos distintivos para cada URIMAP:

| Función             | Nombre de URIMAP | PATH                      | WEBSERVICE |
|---------------------|------------------|---------------------------|------------|
| INQUIRE SINGLE ITEM | INQSURI          | /exampleApp/inquireSingle | EXINQSWWS  |
| PLACE ORDER         | ORDRURI          | /exampleApp/placeOrder    | EXORDRWS   |

| Función                                  | Nombre de URIMAP | PATH                      | WEBSERVICE |
|------------------------------------------|------------------|---------------------------|------------|
| Punto final de DISPATCH STOCK (opcional) | ODEPURI          | /exampleApp/dispatchOrder | EXODEPWS   |

### Finalización de la instalación:

Para completar la instalación, instale el grupo RDO que contiene las definiciones de recursos.

### Procedimiento

Pulse el botón derecho del grupo de recursos en la ventana Definiciones del grupo de recursos. Seleccione **Instalar**. Asegúrese de que el CICSplex es correcto y que selecciona la región de destino, a continuación, pulse **Aceptar**.

### Resultados

Su RDO está instalado y la aplicación está preparada para utilizarse.

Opcional: los usuarios que no utilizan CICS Explorer pueden utilizar la transacción CEDA para corregir e instalar los recursos contenidos en los grupos DFH\$EXBS y DFH\$EXWS.

## Configuración del cliente web

Antes de poder utilizar el cliente web, debe desplegar el archivo web de Java (WAR) para el cliente web inicial en uno de los entornos soportados y configurarlo para llamar a los puntos finales adecuados del sistema CICS.

### Acerca de esta tarea

Los siguientes entornos se soportan para las dos versiones de la aplicación frontal de cliente web ExampleAppClientV855.war y ExampleAppWrapperClientV855.war:

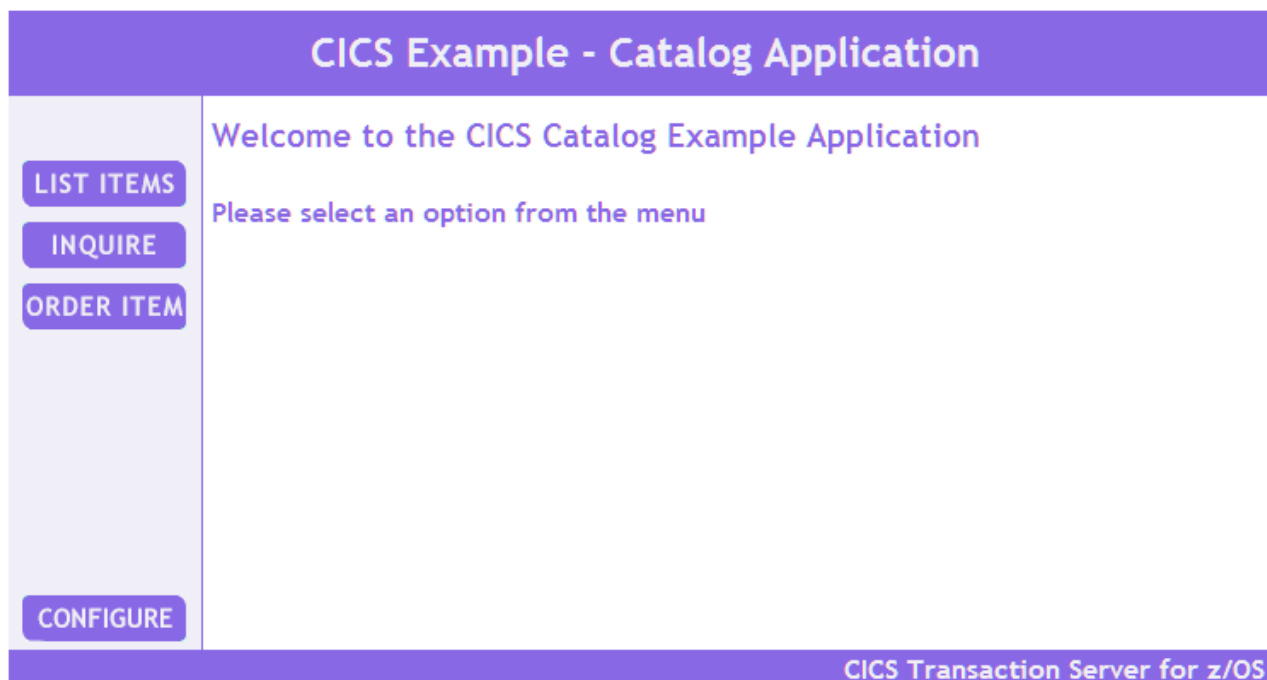
- Servidor JVM de CICS Liberty con el perfil de WebSphere Liberty más reciente

Los archivos WAR están ubicados en el directorio *hlq/samples/webservices/client* en z/OS UNIX.

### Procedimiento

1. Para iniciar el cliente web, entre la siguiente URL en el navegador web, donde *mylibertyserver* es el nombre de host del servidor Liberty JVM en el que está instalado el cliente web.
  - Para ExampleAppClientV855.war, utilice la URL `http://mylibertyserver:9080/ExampleAppClientV855/`
  - Para ExampleAppWrapperClientV855.war, utilice la URL `http://mylibertyserver:9080/ExampleAppWrapperClientV855/`

La aplicación de ejemplo visualizará esta página:



2. Pulse **CONFIGURE** para visualizar la página de configuración. Se visualizará la página de configuración.
3. Especifique el nuevo punto final para el servicio web Consultar catálogo, Consultar elemento y Efectuar pedido.
  - a. En las URL sustituya la serie `myCicsServer` con el nombre del sistema en el que se está ejecutando CICS.
  - b. Sustituya el número de puerto `8080` por el número de puerto configurado en el recurso de definición `TCPIP SERVICE`.
4. Pulse **SUBMIT**.

## Resultados

La aplicación web está ahora preparada para ejecutarse.

## Qué hacer a continuación

La URL para la invocación de servicios web está almacenada en una sesión HTTP. Por lo tanto, es necesario repetir este paso de configuración cada vez que un navegador web se conecta por primera vez al cliente.

## Ejecución de la aplicación habilitada para el servicio web

Puede invocar la aplicación de ejemplo desde un navegador web.

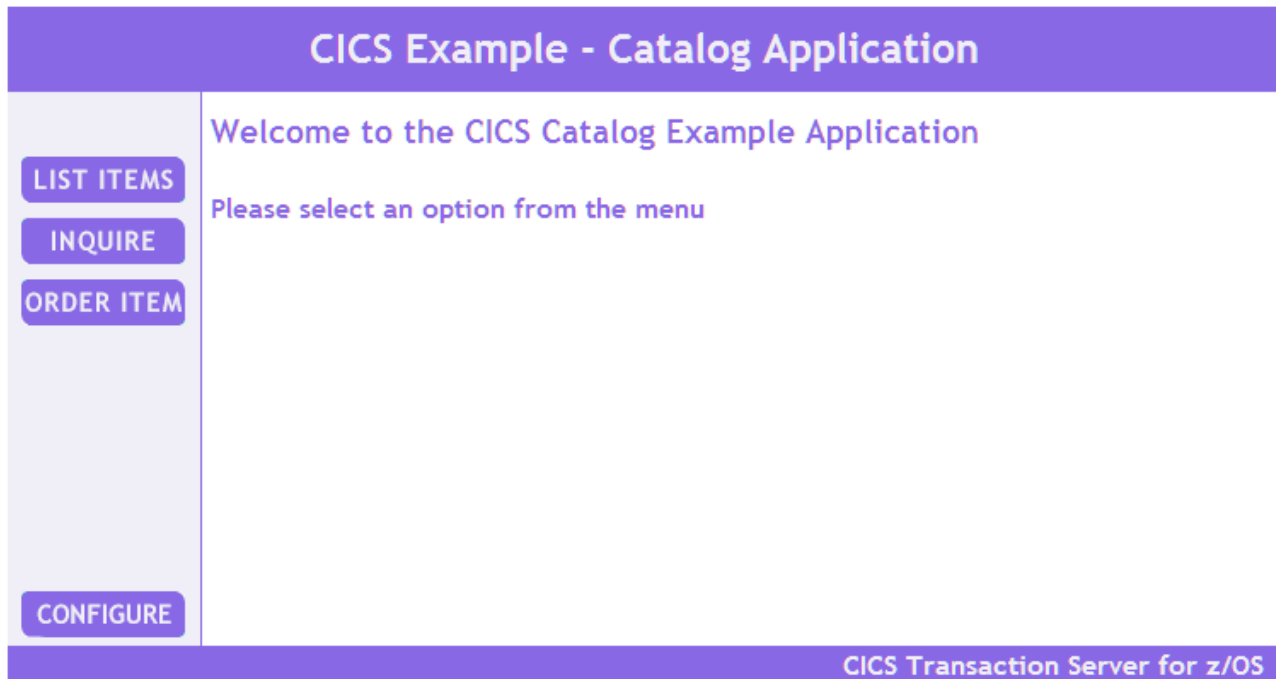
## Acerca de esta tarea

Asegúrese de que ha configurado el cliente web antes de continuar. Consulte *Configuring the example application*

## Procedimiento

1. Entre la siguiente URL en el navegador web: `http://mylibertyserver:9080/ExampleAppClientV855/`, donde *mylibertyserver* es el nombre de host del

servidor en el que se ha instalado el cliente de servicio web. La aplicación de ejemplo visualizará esta página:



2. Pulse el botón **CONSULTA**. La aplicación de ejemplo visualizará esta página:

INQUIRE

ORDER ITEM

BACK

CONFIGURE

CICS Example - Catalog Application

Enter Catalog Item Reference Number

Start List From Item Number0010

SUBMIT

CICS Transaction Server for z/OS

3. Especifique un número de artículo y pulse el botón **SOMETER**.

**Consejo:** La aplicación base se prepara con números de elementos en la secuencia 0010, 0020, ... a 0210.  
La aplicación visualizará la página siguiente, que contiene una lista de elementos del catálogo a partir del número de artículo que ha especificado.



CICS Example - Catalog Application

LIST ITEMS

INQUIRE

ORDER ITEM

BACK

CONFIGURE

Item Details - Select Item to Place Order

| Item | Description                              | In Stock | On Order | Cost    | Select                           |
|------|------------------------------------------|----------|----------|---------|----------------------------------|
| 0010 | Ball Pens Black 24pk                     | 13       | 0        | £2.90   | <input type="radio"/>            |
| 0020 | Ball Pens Blue 24pk                      | 2        | 50       | £2.90   | <input type="radio"/>            |
| 0030 | Ball Pens Red 24pk                       | 38       | 0        | £2.90   | <input type="radio"/>            |
| 0040 | Ball Pens Green 24pk                     | 71       | 0        | £2.90   | <input checked="" type="radio"/> |
| 0050 | Pencil with eraser 12pk                  | 70       | 0        | £1.78   | <input type="radio"/>            |
| 0060 | Highlighters Assorted 5pk                | 11       | 40       | £3.89   | <input type="radio"/>            |
| 0070 | Laser Paper 28-lb 108 Bright 500/ream    | 90       | 20       | £7.44   | <input type="radio"/>            |
| 0080 | Laser Paper 28-lb 108 Bright 2500/case   | 25       | 0        | £33.54  | <input type="radio"/>            |
| 0090 | Blue Laser Paper 20lb 500/ream           | 22       | 0        | £5.35   | <input type="radio"/>            |
| 0100 | Green Laser Paper 20lb 500/ream          | 3        | 20       | £5.35   | <input type="radio"/>            |
| 0110 | IBM Network Printer 24 - Toner cart      | 8        | 0        | £169.56 | <input type="radio"/>            |
| 0120 | Standard Diary: Week to view 8 1/4x5 3/4 | 7        | 0        | £25.99  | <input type="radio"/>            |
| 0130 | Wall Planner: Eraseable 36x24            | 3        | 0        | £18.85  | <input type="radio"/>            |
| 0140 | 70 Sheet Hard Back wire bound notepad    | 84       | 0        | £5.89   | <input type="radio"/>            |
| 0150 | Sticky Notes 3x3 Assorted Colors 5pk     | 22       | 45       | £5.35   | <input type="radio"/>            |

SUBMIT

CICS Transaction Server for z/OS

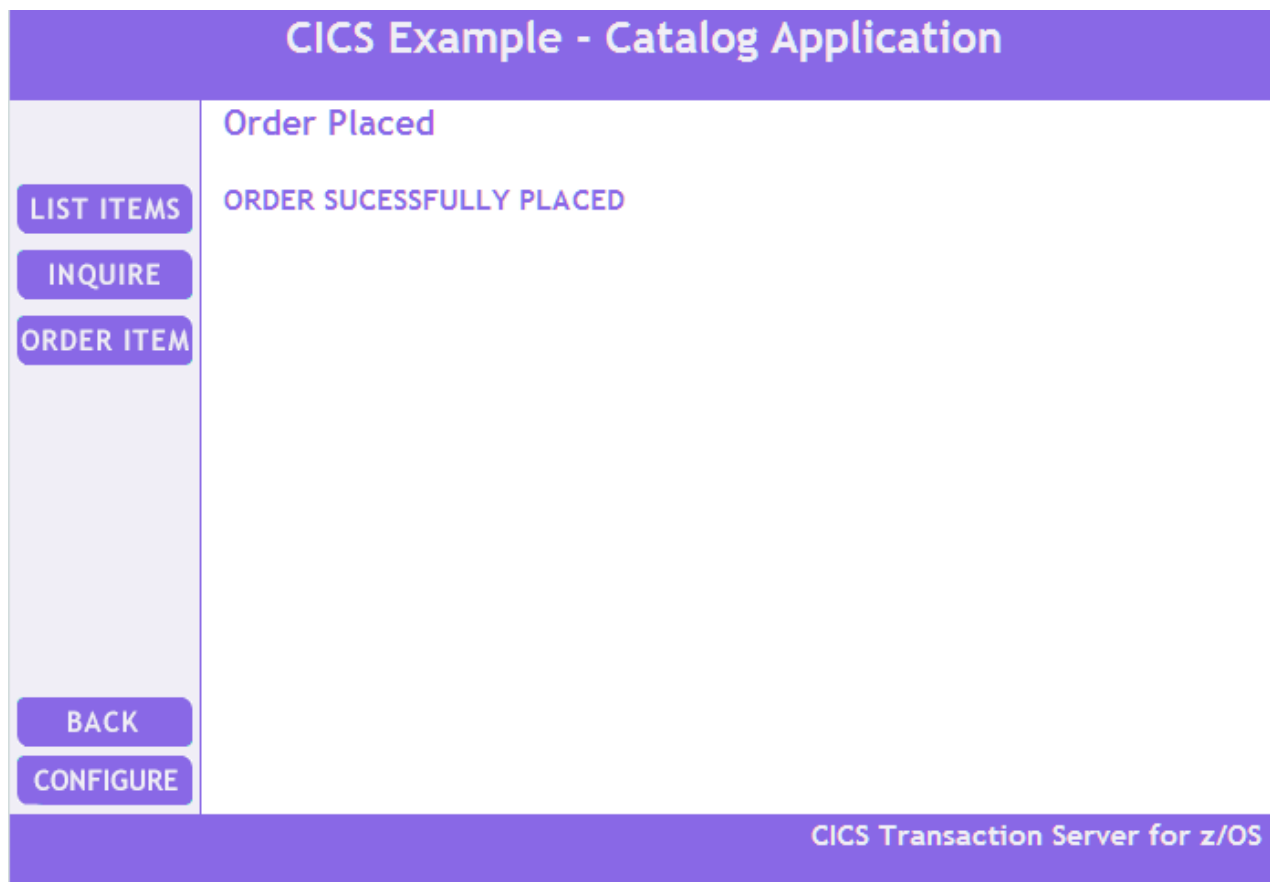
4. Seleccione el artículo que desea solicitar.
  - a. Pulse el botón de selección de la columna **Seleccionar** correspondiente al elemento que desea solicitar.
  - b. Pulse el botón **SOMETER**.

La aplicación visualizará esta página:

| CICS Example - Catalog Application |                       |       |
|------------------------------------|-----------------------|-------|
|                                    | Enter Order Details   |       |
| LIST ITEMS                         | Item Reference Number | 0040  |
| INQUIRE                            | Quantity              | 001   |
|                                    | User Name             | AUSER |
|                                    | Department Name       | CICS1 |
|                                    | SUBMIT                |       |
| BACK                               |                       |       |
| CONFIGURE                          |                       |       |
| CICS Transaction Server for z/OS   |                       |       |

5. Para efectuar un pedido, especifique la información siguiente.
- Complete el campo **Cantidad**. Especifique el número de artículos que desea solicitar.
  - Complete el campo **Nombre de usuario**. Entre una serie de 1 a 8 caracteres. La aplicación base no comprueba el valor especificado aquí.
  - Complete el campo **Nombre de departamento**. Entre una serie de 1 a 8 caracteres. La aplicación base no comprueba el valor especificado aquí.
  - Pulse el botón **SOMETER**.

La aplicación visualiza la página siguiente para confirmar que el pedido se ha efectuado:



## Despliegue de la aplicación de ejemplo

Puede utilizar el asistente de servicios web para desplegar partes de la aplicación de ejemplo como un servicio web. Aunque la aplicación funciona sin realizar esta tarea, debe realizar una tarea similar si desea desplegar sus propias aplicaciones para ampliar la aplicación de ejemplo.

### Extracción de la interfaz de programa

Para desplegar un programa con el asistente de servicio web de CICS, debe crear un libro de copias que coincida con COMMAREA o la interfaz de contenedor.

### Acerca de esta tarea

En este ejemplo, la función INQUIRE SINGLE ITEM del programa de gestor de catálogos central (DFH0XCMN) se despliega como servicio web. La interfaz para este programa es un COMMAREA; la estructura de COMMAREA se define en el libro de copias DFH0XCP1:

\* Estructura

COMMAREA de catálogo

```
03 CA-REQUEST-ID PIC X(6).
03 CA-RETURN-CODE PIC 9(2).
03 CA-RESPONSE-MESSAGE PIC X(79).
03 CA-REQUEST-SPECIFIC PIC X(911).
```

\* Campos utilizados en Inquire Catalog (consulta de catálogo)

```
03 CA-INQUIRE-REQUEST REDEFINES CA-REQUEST-SPECIFIC.
05 CA-LIST-START-REF PIC 9(4).
05 CA-LAST-ITEM-REF PIC 9(4).
```

```

05 CA-ITEM-COUNT PIC 9(3).
05 CA-INQUIRY-RESPONSE-DATA PIC X(900).
05 CA-CAT-ITEM REDEFINES CA-INQUIRY-RESPONSE-DATA
 OCCURS 15 TIMES.
 07 CA-ITEM-REF PIC 9(4).
 07 CA-DESCRIPTION PIC X(40).
 07 CA-DEPARTMENT PIC 9(3).
 07 CA-COST PIC X(6).
 07 IN-STOCK PIC 9(4).
 07 ON-ORDER PIC 9(3).
* Campos utilizados en Inquire Single (consulta única)
03 CA-INQUIRE-SINGLE REDEFINES CA-REQUEST-SPECIFIC.
05 CA-ITEM-REF-REQ PIC 9(4).
05 FILLER PIC 9(4).
05 FILLER PIC 9(3).
05 CA-SINGLE-ITEM.
 07 CA-SNGL-ITEM-REF PIC 9(4).
 07 CA-SNGL-DESCRIPTION PIC X(40).
 07 CA-SNGL-DEPARTMENT PIC 9(3).
 07 CA-SNGL-COST PIC X(6).
 07 IN-SNGL-STOCK PIC 9(4).
 07 ON-SNGL-ORDER PIC 9(3).
05 FILLER PIC X(840).
* Campos utilizados en Place Order (efectuar pedido)
03 CA-ORDER-REQUEST REDEFINES CA-REQUEST-SPECIFIC.
05 CA-USERID PIC X(8).
05 CA-CHARGE-DEPT PIC X(8).
05 CA-ITEM-REF-NUMBER PIC 9(4).
05 CA-QUANTITY-REQ PIC 9(3).
05 FILLER PIC X(888).

```

El libro de copias define tres interfaces individuales para las funciones INQUIRE CATALOG, INQUIRE SINGLE ITEM y PLACE ORDER, que se superponen una sobre otra en el libro de copias. Sin embargo, el programa de utilidad DFHLS2WS no soporta la sentencia REDEFINES. Por lo tanto, debe extraer del libro de copias combinado sólo aquellas secciones que están relacionadas con la función inquire single:

\* Estructura COMMAREA de catálogo

```

03 CA-REQUEST-ID PIC X(6).
03 CA-RETURN-CODE PIC 9(2) DISPLAY.
03 CA-RESPONSE-MESSAGE PIC X(79).
* Campos utilizados en Inquire Single (consulta única)
03 CA-INQUIRE-SINGLE.
05 CA-ITEM-REF-REQ PIC 9(4) DISPLAY.
05 FILLER PIC X(4) DISPLAY.
05 FILLER PIC X(3) DISPLAY.
05 CA-SINGLE-ITEM.
 07 CA-SNGL-ITEM-REF PIC 9(4) DISPLAY.
 07 CA-SNGL-DESCRIPTION PIC X(40).
 07 CA-SNGL-DEPARTMENT PIC 9(3) DISPLAY.
 07 CA-SNGL-COST PIC X(6).
 07 IN-SNGL-STOCK PIC 9(4) DISPLAY.
 07 ON-SNGL-ORDER PIC 9(3) DISPLAY.
05 FILLER PIC X(840).

```

se ha eliminado y se ha sustituido por la sección del libro de copias que se ha redefinido para la función inquire single. El libro de copias no es adecuado para su uso con el asistente de servicios web.

El libro de copias se proporciona con la aplicación de ejemplo como libro de copias DFH0XCP4.

## Ejecución del programa del asistente de servicios web DFHLS2WS

El asistente de servicios web de CICS consta de dos programas por lotes que pueden ayudarle a transformar las aplicaciones CICS existentes en servicios web, y a permitir que las aplicaciones CICS utilicen servicios web proporcionados por proveedores externos. El programa DFHLS2WS transforma una estructura de lenguaje para generar un archivo de enlace de servicio web y una descripción de servicio web.

### Procedimiento

1. Copie el JCL de ejemplo proporcionado a un archivo de trabajo adecuado. El JCL se proporciona en `samples/webservices/JCL/LS2WS`.
2. Añada una tarjeta JOB válida al JCL.
3. Codifique los parámetros para DFHLS2WS. Los siguientes parámetros son necesarios para la función INQUIRE SINGLE ITEM de la aplicación de ejemplo:

```
//INPUT.SYSUT1 DD *
LOGFILE=/u/exampleapp/wsbinding/inquireSingle.log
PDSLIB=CICSHLQ.SDFHSAMP
REQMEM=DFH0XCP4
RESPMEM=DFH0XCP4
LANG=COBOL
PGMNAME=DFH0XCMN
PGMINT=COMMAREA
URI=mycicsserver:myport/exampleApp/inquireSingle
WSBIND=/u/exampleapp/wsbinding/inquireSingle.wsbinding
WSDL=/u/exampleapp/wsd1/inquireSingle.wsd1
*/
```

#### **LOGFILE=/u/exampleapp/wsbinding/inquireSingle.log**

El archivo que se utiliza para registrar información de diagnóstico de DFHLS2WS. El archivo normalmente sólo lo utiliza la organización de soporte de software IBM.

#### **PDSLIB=CICSHLQ.SDFHSAMP**

El nombre del conjunto de datos particionados (PDS) donde el asistente de servicios web busca los libros de copias que definen las estructuras de solicitud y respuesta. En el ejemplo, es el conjunto de datos instalado de CICS, SDFHSAMP.

#### **REQMEM=DFH0XCP4**

#### **RESPMEM=DFH0XCP4**

Estos parámetros definen la estructura de lenguaje para la solicitud y la respuesta para el programa. En el ejemplo, la solicitud y la respuesta tienen la misma estructura y se definen por el mismo libro de copias.

#### **LANG=COBOL**

El programa destino y las estructuras de datos se escriben en COBOL.

#### **PGMNAME=DFH0XCMN**

El nombre del programa destino que se inicia cuando se recibe una solicitud de servicio web.

#### **PGMINT=COMMAREA**

El programa destino se invoca con una interfaz COMMAREA.

#### **URI=mycicsserver:myport/exampleApp/inquireSingle**

La parte exclusiva del URI que se utiliza en la definición de servicio web generada, y para crear el recurso URIMAP que correlaciona solicitudes de entrada con el servicio web correcto. El valor especificado da como resultado que el servicio esté disponible para clientes externos en:

`http://mycicsserver:myport/exampleApp/inquireSingle`

donde *mycicsserver* y *myport* son la dirección de servidor y puerto de CICS en el que se ha instalado este recurso WEBSERVICE.

**Nota:** El parámetro no tiene una '/' de inicio.

#### **WSBIND=/u/exampleapp/wsbind/inquireSingle.wsbind**

La ubicación en z/OS UNIX en la que se escribe el archivo de enlace de servicio web.

**Nota:** Si el archivo se va a utilizar con el mecanismo de exploración de interconexión debe tener la extensión `.wsbind`.

#### **WSDL=/u/exampleapp/wsd1/inquireSingle.wsd1**

La ubicación en z/OS UNIX en la que se escribe el archivo que contiene la descripción de servicio web generada. Es una buena práctica utilizar nombres coincidentes para el archivo de enlace de servicio web y su correspondiente descripción de servicio web.

De forma convencional, los archivos que contienen las descripciones de servicio web tienen la extensión `.wsdl`.

La descripción de servicios web proporciona la información que debe utilizar un cliente para acceder al servicio web. Contiene una definición de esquema XML de la solicitud y la respuesta, y la información de ubicación para el servicio.

4. Ejecute el trabajo. Se crean una descripción de servicio web y un archivo de enlace de servicio web en las ubicaciones especificadas.

### **Ejemplo del documento WSDL generado**

Un ejemplo del documento de descripción de servicio web (WSDL) generado cuando se está ejecutando el programa del asistente de servicios web DFHLS2WS.

```
<?xml version="1.0" ?>
<definitions targetNamespace="http://www.DFH0XCMN.DFH0XCP4.com" xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:reqns="http://www.DFH0XCMN.DFH0XCP4.Request.com" xmlns:resns="http://www.DFH0XCMN.DFH0XCP4.Response.com"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://www.DFH0XCMN.DFH0XCP4.com">
 <types>
 <xsd:schema attributeFormDefault="qualified" elementFormDefault="qualified"
targetNamespace="http://www.DFH0XCMN.DFH0XCP4.Request.com" xmlns:tns="http://www.DFH0XCMN.DFH0XCP4.Request.com"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:complexType abstract="false" block="#all" final="#all" mixed="false" name="ProgramInterface">
 <xsd:annotation>
 <xsd:documentation source="http://www.ibm.com/software/http/cics/annotations">
 This schema was generated by the CICS web services assistant.
 </xsd:documentation>
 </xsd:annotation>
 <xsd:sequence>
 <xsd:element name="ca_request_id" nillable="false">
 <xsd:simpletype>
 <xsd:annotation>
 <xsd:appinfo source="http://www.ibm.com/software/http/cics/annotations">
 #Thu Nov 03 11:55:26 GMT 2005 com.ibm.cics.wsd1.properties.synchronized=false
 </xsd:appinfo>
 </xsd:annotation>
 <xsd:restriction base="xsd:string">
 <xsd:maxLength value="6"/>
 <xsd:whitespace value="preserve"/>
 </xsd:restriction>
 </xsd:simpletype>
 </xsd:element>
 most of the schema for the request is removed
 </xsd:sequence>
 </xsd:complexType>
 <xsd:element name="DFH0XCMNOperation" nillable="false" type="tns:ProgramInterface"/>
 </xsd:schema>
```

```

<xsd:schema attributeFormDefault="qualified" elementFormDefault="qualified"
targetNamespace="http://www.DFH0XCMN.DFH0XCP4.Response.com" xmlns:tns="http://www.DFH0XCMN.DFH0XCP4.Response.com"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">

... schema content for the reply is removed

</xsd:schema>
</types>
<message name="DFH0XCMNOperationResponse">
 <part element="resns:DFH0XCMNOperationResponse" name="ResponsePart"/>
</message>
<message name="DFH0XCMNOperationRequest">
 <part element="reqns:DFH0XCMNOperation" name="RequestPart"/>
</message>
<porttype name="DFH0XCMNPort">
 <operation name="DFH0XCMNOperation">
 <input message="tns:DFH0XCMNOperationRequest" name="DFH0XCMNOperationRequest"/>
 <output message="tns:DFH0XCMNOperationResponse" name="DFH0XCMNOperationResponse"/>
 </operation>
</porttype>
<binding name="DFH0XCMNHTTPSoapBinding" type="tns:DFH0XCMNPort">
 <!-- This soap:binding indicates the use of SOAP 1.1 -->
 <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
 <!-- This soap:binding indicates the use of SOAP 1.2 -->
 <!-- <soap:binding style="document" transport="http://www.w3.org/2003/05/soap-http"/> -->
 <operation name="DFH0XCMNOperation">
 <soap:operation soapAction="" style="document"/>
 <input name="DFH0XCMNOperationRequest">
 <soap:body parts="RequestPart" use="literal"/>
 </input>
 <output name="DFH0XCMNOperationResponse">
 <soap:body parts="ResponsePart" use="literal"/>
 </output>
 </operation>
</binding>
<service name="DFH0XCMNService">
 <port binding="tns:DFH0XCMNHTTPSoapBinding" name="DFH0XCMNPort">
 <!-- This soap:address indicates the location of the web service over HTTP.
 Please replace "my-server" with the TCP/IP host name of your CICS region.
 Please replace "my-port" with the port number of your CICS TCPIP SERVICE. -->
 <soap:address location="http://my-server:my-port/exampleApp/inquireSingles.log"/>
 <!-- This soap:address indicates the location of the web service over HTTPS. -->
 <!-- <soap:address location="https://my-server:my-port/exampleApp/inquireSingles.log"/> -->
 <!-- This soap:address indicates the location of the web service over Websphere MQSeries.
 Please replace "my-queue" with the appropriate queue name. -->
 <!-- <soap:address location="jms:/queue?destination=my-queue&connectionFactory=()&targetService=/exampleApp/inquireSingles.log&initialContextFactory=com.ibm.mq.jms.NoJndi" /> -->
 </port>
</service>
</definitions>

```

## Despliegue el archivo de enlaces de servicio web

El archivo de enlace WEBSERVICE, creado por DFHLS2WS, se despliega en la región de CICS de forma dinámica cuando instala un recurso PIPELINE.

## Acerca de esta tarea

Cuando se emite un mandato de exploración de interconexión, CICS explora el directorio de recogida para buscar archivos de enlace WEBSERVICE con la extensión .wsbind. Para cada archivo de enlace encontrado, CICS determina si instalar un recurso WEBSERVICE.

También se crea un recurso URIMAP para correlacionar el URI, como se proporciona en el JCL, para el recurso WEBSERVICE instalado y PIPELINE en la que está instalada el servicio web. Cuando se descarta un recurso WEBSERVICE explorado, el recurso URIMAP asociado a él también se descarta.

## Procedimiento

1. Modifique la definición PIPELINE para la interconexión de proveedor PIPELINE(EXPIPE01) en CICS Explorer seleccionando **Definiciones** >

**Definiciones de interconexión.** Efectúe doble pulsación en EXPIPE01 para abrir el editor Definición de interconexión (EXPIPE01). En la pestaña **Atributos**, cambie el parámetro **Directorio WS** por /u/exampleapp/wsbind. El directorio de recogida contiene el archivo de enlace WEBSERVICE que ha generado con DFHLS2WS.

2. Copie cualquier otro archivo de enlace WEBSERVICE utilizado por la aplicación para el mismo directorio. En este ejemplo, se copian los siguientes archivos:

```
inquireCatalog
placeOrder
```

Se proporcionan en el directorio /usr/lpp/cicsts/samples/webservices/wsbind/provider.

3. Instale el recurso PIPELINE.

## Resultados

CICS crea dos recursos URIMAP; la primera definición URIMAP se necesita en un proveedor de servicios cuando contiene información que se correlaciona con el URI de una solicitud de servicio web de entrada a otros recursos (como el recurso PIPELINE) que da servicio a la solicitud. El segundo URIMAP contiene información que correlaciona el URI de una solicitud de entrada para el documento o documentos WSDL asociados al servicio web.

## Componentes de la aplicación base

Utilice estas tablas para entender los componentes de la aplicación base y los miembros proporcionados en el ejemplo SDFHSAMP. Los miembros de SDFHSAMP listados contienen correlaciones BMS, origen COBOL y libros de copias para la aplicación base, la aplicación de cliente de servicio web y los módulos de derivador.

*Tabla 18. Miembros de SDFHSAMP que contienen correlaciones BMS*

| Nombre de miembro | Descripción                                                                                                                                                                                                                      |
|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DFH0XS1           | Las macros BMS para el conjunto de correlaciones que consta de la correlación (EXMENU) para la pantalla <b>Menú principal</b> y la correlación (EXORDR) para la pantalla <b>Detalles del pedido</b> .                            |
| DFH0XS2           | Las macros BMS para el conjunto de correlaciones constan de la correlación (EXINQC) para la pantalla <b>Consultar catálogo</b> .                                                                                                 |
| DFH0XS3           | Las macros BMS para el conjunto de correlaciones constan de la correlación (EXCONF) para la pantalla <b>Configurar la aplicación de catálogo de ejemplo de CICS</b> .                                                            |
| DFH0XM1           | El libro de copias COBOL generado ensamblando DFH0XS1. DFH0XGUI y DFH0XCUI incluyen este libro de copias                                                                                                                         |
| DFH0XM2U          | Libro de copias COBOL generado ensamblando DFH0XS2 y editando el resultado para incluir una estructura de matriz indexada para facilitar la programación del libro de copias. DFH0XGUI y DFH0XCUI incluyen este libro de copias. |
| DFH0XM3           | El libro de copias COBOL generado ensamblando DFH0XS3. DFH0XCFG incluye este libro de copias                                                                                                                                     |



Tabla 19. Miembros de SDFHSAMP que contienen el origen COBOL para la aplicación base

| Nombre de miembro | Descripción                                                                                                                                                                                                                                                                    |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DFH0XCFG          | Programa invocado por la transacción ECFG para leer y actualizar el archivo de configuración VSAM.                                                                                                                                                                             |
| DFH0XCMN          | Programa de controlador para la aplicación de catálogo. Todas las solicitudes pasan a través del programa de controlador.                                                                                                                                                      |
| DFH0XGUI          | Programa invocado por la transacción EGUI para gestionar el envío de correlaciones BMS al usuario de terminal y la recepción de correlaciones del usuario de terminal. Este programa enlaza con el programa DFH0XCMN.                                                          |
| DFH0XODE          | Una de las dos versiones del punto final para el servicio web de asignación de pedido. Esta es la versión que se ejecuta en CICS. Este programa establece el texto "Pedido en la asignación" en el COMMAREA de vuelta.                                                         |
| DFH0XSDS          | Un <i>apéndice</i> o versión ficticia del programa de almacén de datos que permite a la aplicación trabajar cuando el archivo de catálogo VSAM no se ha configurado. DFH0XSDS utiliza los datos definidos en el programa en lugar de los datos almacenados en un archivo VSAM. |
| DFH0XSOD          | Una versión de apéndice del programa de asignación de pedidos. Establece el código de retorno de COMMAREA en 0 y lo devuelve a su interlocutor. DFH0XSOD se utiliza cuando no se necesitan servicios web de salida.                                                            |
| DFH0XSSM          | Una versión de apéndice del programa de gestor de stock (reaprovisionamiento). DFH0XSSM establece el código de retorno en COMMAREA en 0 y lo devuelve a su interlocutor.                                                                                                       |
| DFH0XVDS          | La versión VSAM del programa de almacén de datos. DFH0XVDS accede al archivo VSAM para llevar a cabo lecturas y actualizaciones del catálogo.                                                                                                                                  |
| DFH0XWOD          | La versión de servicio web del programa de asignación de pedidos. DFH0XWOD emite un EXEC CICS INVOKE WEBSERVICE para que el servicio web de salida invoque un asignador de pedidos.                                                                                            |

Tabla 20. Miembros de SDFHSAMP que contienen libros de copias COBOL para la aplicación base

| Nombre de miembro | Descripción                                                                                                                                                                                                                                                                  |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DFH0XCP1          | Define una estructura COMMAREA que incluye la solicitud y respuesta para las funciones inquire catalog, inquire single y place order. Los programas DFH0XCMN, DFH0XCUI, DFH0XECC, DFH0XGUI, DFH0XICW, DFH0XISW, DFH0XPOW, DFH0XSDS y DFH0XVDS incluyen este libro de copias. |
| DFH0XCP2          | Define una estructura COMMAREA para el asignador de pedidos y los módulos de gestor de stock. Los programas DFH0XCMN, DFH0XSOD, DFH0XSSM y DFH0XWOD incluyen este libro de copias.                                                                                           |
| DFH0XCP3          | Define una estructura de datos para una solicitud y respuesta de la función inquire catalog. Se utiliza como entrada para DFHLS2WS para generar inquireCatalog.wsdl y inquireCatalog.wsbind.                                                                                 |
| DFH0XCP4          | Define una estructura de datos para una solicitud y respuesta de la función inquire single. Se utiliza como entrada para DFHLS2WS para generar inquireSingle.wsdl y inquireSingle.wsbind.                                                                                    |

*Tabla 20. Miembros de SDFHSAMP que contienen libros de copias COBOL para la aplicación base (continuación)*

| Nombre de miembro | Descripción                                                                                                                                                                               |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DFH0XCP5          | Define una estructura de datos para una solicitud y respuesta de la función place order. Se utiliza como entrada para DFHLS2WS para generar placeOrder.wsdl y placeOrder.wsbind.          |
| DFH0XCP6          | Define una estructura de datos para una solicitud y respuesta de la función dispatch order. Se utiliza como entrada para DFHLS2WS para generar dispatchOrder.wsdl y dispatchOrder.wsbind. |
| DFH0XCP7          | Define una estructura de datos para una solicitud de la función dispatch order. Los programas DFH0XODE y DFH0XWOD incluyen este libro de copias                                           |
| DFH0XCP8          | Define una estructura de datos para una respuesta de la función dispatch order. Los programas DFH0XODE y DFH0XWOD incluyen este libro de copias.                                          |

*Tabla 21. Miembros SDFHSAMP que contienen el código de origen COBOL para la aplicación de cliente de servicio web que se ejecuta en CICS*

| Nombre de miembro | Descripción                                                                                                                                                                                                                                          |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DFH0XCUI          | Programa invocado por la transacción ECLI para gestionar el envío de correlaciones BMS al usuario de terminal y la recepción de correlaciones del usuario de terminal. Enlaza con el programa DFH0XECC.                                              |
| DFH0XECC          | Crea las solicitudes de servicio web de salida para la aplicación base, utilizando el mandato EXEC CICS INVOKE WEBSERVICE. El servicio web especificado es uno de los siguientes:<br>inquireCatalogClient<br>inquireSingleClient<br>placeOrderClient |

*Tabla 22. Miembros de SDFHSAMP que contienen libros de copias COBOL para la aplicación de cliente de servicio web que se ejecuta en CICS. Todos están generados por DFHWS2LS, e incluidos por el programa DFH0XECC.*

| Nombre de miembro | Descripción                                                                      |
|-------------------|----------------------------------------------------------------------------------|
| DFH0XCPA          | Define una estructura de datos para una solicitud de la función inquire catalog. |
| DFH0XCPB          | Define una estructura de datos para una respuesta de la función inquire catalog. |
| DFH0XCPC          | Define una estructura de datos para una solicitud de la función inquire single.  |
| DFH0XCPD          | Define una estructura de datos para una respuesta de la función inquire single.  |
| DFH0XCPE          | Define una estructura de datos para una solicitud de la función place order.     |
| DFH0XCPF          | Define una estructura de datos para una respuesta de la función place order.     |

Tabla 23. Miembros de SDFHSAMP que contienen el código de origen COBOL para módulos de derivador

| Nombre de miembro | Descripción                                        |
|-------------------|----------------------------------------------------|
| DFH0XECC          | Programa del cliente de servicios web              |
| DFH0XICW          | Programa derivado para el servicio inquireCatalog. |
| DFH0XISW          | Programa derivado para el servicio inquireSingle.  |
| DFH0XPOW          | Programa derivado para el servicio purchaseOrder.  |

Tabla 24. Miembros de SDFHSAMP que contienen libros de copias COBOL para módulos de derivador

| Nombre de miembro | Descripción                                                                                                                         |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------|
| DFH0XWC1          | Define una estructura de datos para una solicitud de la función inquire catalog. El programa DFH0XICW incluye este libro de copias. |
| DFH0XWC2          | Define una estructura de datos para una respuesta de la función inquire catalog. El programa DFH0XICW incluye este libro de copias. |
| DFH0XWC3          | Define una estructura de datos para una solicitud de la función inquire single. El programa DFH0XISW incluye este libro de copias.  |
| DFH0XWC4          | Define una estructura de datos para una respuesta de la función inquire single. El programa DFH0XISW incluye este libro de copias.  |
| DFH0XWC5          | Define una estructura de datos para una solicitud de la función place order. El programa DFH0XPOW incluye este libro de copias.     |
| DFH0XWC6          | Define una estructura de datos para una respuesta de la función place order. El programa DFH0XPOW incluye este libro de copias      |

Tabla 25. Definiciones de recursos CICS

| Nombre del recurso | Tipo de recurso                       | Comentario                                                                                                                  |
|--------------------|---------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| EXAMPLE            | Grupo de definiciones de recurso CICS | Definiciones de recursos CICS necesarias para la aplicación de ejemplo.                                                     |
| EGUI               | TRANSACTION                           | Transacción para invocar el programa DFH0XGUI para iniciar la interfaz BMS para la aplicación (personalizable).             |
| ECFG               | TRANSACTION                           | Transacción para invocar el programa DFH0XCFG para iniciar la interfaz BMS de la configuración de ejemplo (personalizable). |
| EXMPCAT            | FILE                                  | Definición de archivo del archivo EXMPCAT VSAM para el catálogo de aplicaciones (personalizable).                           |
| EXMPCONF           | FILE                                  | Definición del archivo de configuración de la aplicación EXMPCONF                                                           |

## El programa gestor de catálogos

El gestor de catálogos es el programa controlador de la lógica comercial de la aplicación de ejemplo, y todas las interacciones con la aplicación de ejemplo pasan a través de él.

Para asegurarse de que la lógica de programa es simple, la comprobación de tipo y la recuperación de errores que realiza el gestor de catálogos es limitada.

El gestor de catálogos da soporte a diversas operaciones. Los parámetros de entrada y salida de cada operación se definen en una sola estructura de datos, que se pasa al programa y desde él en una COMMAREA.

### Estructuras COMMAREA:

Los datos se pasan entre los programas de cliente y servidor de ejemplo utilizando un área de comunicaciones CICS estándar (COMMAREA).

El siguiente extracto de código muestra la estructura COMMAREA de la aplicación del gestor de catálogos.

```
* Estructura COMMAREA de catálogo
03 CA-REQUEST-ID PIC X(6).
03 CA-RETURN-CODE PIC 9(2).
03 CA-RESPONSE-MESSAGE PIC X(79).
03 CA-REQUEST-SPECIFIC PIC X(911).
* Campos utilizados en Inquire Catalog (consulta de catálogo)
03 CA-INQUIRE-REQUEST REDEFINES CA-REQUEST-SPECIFIC.
05 CA-LIST-START-REF PIC 9(4).
05 CA-LAST-ITEM-REF PIC 9(4).
05 CA-ITEM-COUNT PIC 9(3).
05 CA-INQUIRY-RESPONSE-DATA PIC X(900).
05 CA-CAT-ITEM REDEFINES CA-INQUIRY-RESPONSE-DATA
 OCCURS 15 TIMES.
 07 CA-ITEM-REF PIC 9(4).
 07 CA-DESCRIPTION PIC X(40).
 07 CA-DEPARTMENT PIC 9(3).
 07 CA-COST PIC X(6).
 07 IN-STOCK PIC 9(4).
 07 ON-ORDER PIC 9(3).
* Campos utilizados en Inquire Single (consulta única)
03 CA-INQUIRE-SINGLE REDEFINES CA-REQUEST-SPECIFIC.
05 CA-ITEM-REF-REQ PIC 9(4).
05 FILLER PIC 9(4).
05 FILLER PIC 9(3).
05 CA-SINGLE-ITEM.
 07 CA-SNGL-ITEM-REF PIC 9(4).
 07 CA-SNGL-DESCRIPTION PIC X(40).
 07 CA-SNGL-DEPARTMENT PIC 9(3).
 07 CA-SNGL-COST PIC X(6).
 07 IN-SNGL-STOCK PIC 9(4).
 07 ON-SNGL-ORDER PIC 9(3).
05 FILLER PIC X(840).
* Campos utilizados en Place Order (efectuar pedido)
03 CA-ORDER-REQUEST REDEFINES CA-REQUEST-SPECIFIC.
05 CA-USERID PIC X(8).
05 CA-CHARGE-DEPT PIC X(8).
05 CA-ITEM-REF-NUMBER PIC 9(4).
05 CA-QUANTITY-REQ PIC 9(3).
05 FILLER PIC X(888).

* Estructura COMMAREA de gestor de stocks/expedidor
03 CA-ORD-REQUEST-ID PIC X(6).
```

```

03 CA-ORD-RETURN-CODE PIC 9(2).
03 CA-ORD-RESPONSE-MESSAGE PIC X(79).
03 CA-ORD-REQUEST-SPECIFIC PIC X(23).
* Campos utilizados en Dispatcher (Expedidor)
03 CA-DISPATCH-ORDER REDEFINES CA-ORD-REQUEST-SPECIFIC.
05 CA-ORD-ITEM-REF-NUMBER PIC 9(4).
05 CA-ORD-QUANTITY-REQ PIC 9(3).
05 CA-ORD-USERID PIC X(8).
05 CA-ORD-CHARGE-DEPT PIC X(8).
* Campos utilizados en Stock Manager (gestor de stock)
03 CA-STOCK-MANAGER-UPDATE REDEFINES CA-ORD-REQUEST-SPECIFIC.
05 CA-STK-ITEM-REF-NUMBER PIC 9(4).
05 CA-STK-QUANTITY-REQ PIC 9(3).
05 FILLER PIC X(16).

```

### Códigos de retorno:

Cada operación del gestor de catálogos puede devolver diversos códigos de retorno.

Tabla 26. Códigos de retorno del gestor de catálogos

| Tipo                     | Código | Explicación                                                                                       |
|--------------------------|--------|---------------------------------------------------------------------------------------------------|
| General                  | 00     | Función completada sin errores                                                                    |
| Archivo de catálogo      | 20     | Referencia a artículo no encontrada                                                               |
|                          | 21     | Error al abrir, leer o finalizar el examen del archivo de catálogo                                |
|                          | 22     | Error al actualizar el archivo                                                                    |
| Archivo de configuración | 50     | Error al abrir el archivo de configuración                                                        |
|                          | 51     | El tipo del almacén de datos no es STUB ni VSAM                                                   |
|                          | 52     | El conmutador de servicio web de salida no era Y ni N                                             |
| Servicio web remoto      | 30     | El mandato <b>EXEC CICS INVOKE WEBSERVICE</b> ha devuelto una condición INVREQ                    |
|                          | 31     | El mandato <b>EXEC CICS INVOKE WEBSERVICE</b> ha devuelto una condición NOTFND                    |
|                          | 32     | El mandato <b>EXEC CICS INVOKE WEBSERVICE</b> ha devuelto una condición que no es INVREQ o NOTFND |

Tabla 26. Códigos de retorno del gestor de catálogos (continuación)

| Tipo       | Código | Explicación                                                                                                                                      |
|------------|--------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| Aplicación | 97     | Existencias insuficientes para completar el pedido                                                                                               |
|            | 98     | La cantidad del pedido no es un número positivo                                                                                                  |
|            | 99     | DFH0XCMN ha recibido una COMMAREA en la que el campo CA-REQUEST-ID no está establecido en uno de los siguientes valores: 01INQC, 01INQS o 01ORDR |

### Operación INQUIRE CATALOG:

Esta operación devuelve una lista de hasta 15 artículos de catálogo, a partir del artículo especificado por el llamante.

#### Parámetros de entrada

##### CA-REQUEST-ID

Serie que identifica la operación. En el mandato INQUIRE CATALOG, la serie contiene 01INQC.

##### CA-LIST-START-REF

El número de referencia del primer artículo que debe devolverse.

#### Parámetros de salida

##### CA-RETURN-CODE

Serie que identifica la operación.

##### CA-RESPONSE-MESSAGE

Serie legible por el usuario, que contiene *núm* ITEMS RETURNED, donde *núm* es el número de artículos devueltos.

##### CA-LAST-ITEM-REF

El número de referencia del último artículo devuelto.

##### CA-ITEM-COUNT

Cantidad de artículos devueltos.

##### CA-CAT-ITEM

Una matriz que contiene la lista de artículos de catálogo devueltos. La matriz tiene 15 elementos; si se devuelven menos de 15 elementos, el resto de elementos de la matriz contienen blancos.

### Operación INQUIRE SINGLE ITEM:

Esta operación devuelve un solo artículo de catálogo especificado por el llamante.

#### Parámetros de entrada

##### CA-REQUEST-ID

Serie que identifica la operación. En el mandato INQUIRE SINGLE ITEM, la serie contiene 01INQS.

##### CA-ITEM-REF-REQ

El número de referencia del artículo que debe devolverse.

### Parámetros de salida

#### CA-RETURN-CODE

Serie que identifica la operación.

#### CA-RESPONSE-MESSAGE

Una serie legible que contiene RETURNED ITEM: REF=*item-reference* donde *item-reference* es el número de referencia del artículo devuelto.

#### CA-SINGLE-ITEM

Matriz que contiene en su primer elemento el artículo de catálogo devuelto.

### Operación PLACE ORDER:

Esta operación efectúa un pedido de un solo artículo. Si la cantidad necesaria no está disponible, se devuelve un mensaje al usuario. Si el pedido es satisfactorio, se realiza una llamada al gestor de stocks notificándole qué artículo se ha solicitado y la cantidad solicitada.

### Parámetros de entrada

#### CA-REQUEST-ID

Serie que identifica la operación. En la operación PLACE ORDER, la serie contiene 010RDR.

#### CA-USERID

ID de usuario de 8 caracteres que la aplicación utiliza para el envío y la facturación.

#### CA-CHARGE-DEPT

ID de departamento de 8 caracteres que la aplicación utiliza para el envío y la facturación.

#### CA-ITEM-REF-NUMBER

El número de referencia del artículo que debe solicitarse.

#### CA-QUANTITY-REQ

El número de elementos necesarios.

### Parámetros de salida

#### CA-RETURN-CODE

Serie que identifica la operación.

#### CA-RESPONSE-MESSAGE

Una cadena legible que contiene ORDER SUCCESSFULLY PLACED.

### Operación DISPATCH STOCK:

Esta operación realiza una llamada al programa expedidor de stocks, que a su vez envía el pedido al cliente.

### Parámetros de entrada

#### CA-ORD-REQUEST-ID

Serie que identifica la operación. En la operación DISPATCH ORDER, la serie contiene 01DSP0.

#### CA-ORD-USERID

ID de usuario de 8 caracteres que la aplicación utiliza para el envío y la facturación.

**CA-ORD-CHARGE-DEPT**

ID de departamento de 8 caracteres que la aplicación utiliza para el envío y la facturación.

**CA-ORD-ITEM-REF-NUMBER**

El número de referencia del artículo que debe solicitarse.

**CA-ORD-QUANTITY-REQ**

El número de elementos necesarios.

**Parámetros de salida****CA-ORD-RETURN-CODE**

Serie que identifica la operación.

**Operación NOTIFY STOCK MANAGER:**

Esta operación valora los detalles del pedido que se ha realizado para decidir si es necesario reponer el stock.

**Parámetros de entrada****CA-ORD-REQUEST-ID**

Serie que identifica la operación. Para la operación NOTIFY STOCK MANAGER, la serie contiene 01STK0.

**CA-STK-ITEM-REF-NUMBER**

El número de referencia del artículo que debe solicitarse.

**CA-STK-QUANTITY-REQ**

El número de elementos necesarios.

**Parámetros de salida****CA-ORD-RETURN-CODE**

Serie que identifica la operación.

## Estructuras de archivos y definiciones

La aplicación de ejemplo utiliza dos archivos VSAM: el archivo de catálogo, que contiene los detalles de todos los elementos almacenados y sus niveles de stock, y el archivo de configuración, que contiene las opciones seleccionadas por el usuario para la aplicación.

### Archivo de catálogo

El archivo de catálogo es un archivo KSDS VSAM que contiene toda la información relacionada con el inventario de producto.

### Registros del archivo de catálogo

Los registros del archivo tienen la estructura siguiente:

| Nombre          | Tipo de datos COBOL | Descripción                              |
|-----------------|---------------------|------------------------------------------|
| WS-ITEM-REF-NUM | PIC 9(4)            | Número de referencia de artículo         |
| WS-DESCRIPTION  | PIC X(40)           | Descripción del elemento                 |
| WS-DEPARTMENT   | PIC 9(3)            | Número de identificación de departamento |
| WS-COST         | PIC ZZZ.99          | Precio del artículo                      |



| Nombre      | Tipo de datos COBOL | Descripción                     |
|-------------|---------------------|---------------------------------|
| WS-IN-STOCK | PIC 9(4)            | Número de artículos en stock    |
| WS-ON-ORDER | PIC 9(3)            | Número de artículos bajo pedido |

## Archivo de configuración

El archivo de configuración es un archivo KSDS VSAM que contiene información utilizada para configurar la aplicación de ejemplo.

## Registros del archivo de configuración

El archivo de configuración es un archivo KSDS VSAM con cuatro registros distintos.

*Tabla 27. Registro de información general*

| Nombre         | Tipo de datos COBOL | Descripción                                                                                                                            |
|----------------|---------------------|----------------------------------------------------------------------------------------------------------------------------------------|
| PROGS-KEY      | PIC X(9)            | Campo de clave para el registro de información general que contiene EXMP-CONF.                                                         |
| relleno        | PIC X               |                                                                                                                                        |
| DATASTORE      | PIC X(4)            | Serie de caracteres que especifica el tipo de programa de almacén de datos que debe utilizarse. Los valores son:<br>STUB<br>VSAM       |
| relleno        | PIC X               |                                                                                                                                        |
| DO-OUTBOUND-WS | PIC X               | Un carácter que especifica si el gestor de asignación ha realizado una solicitud de servicio web de salida. Los valores son:<br>S<br>N |
| relleno        | PIC X               |                                                                                                                                        |
| CATMAN-PROG    | PIC X(8)            | El nombre del programa de gestor de catálogos.                                                                                         |
| relleno        | PIC X               |                                                                                                                                        |
| DSSTUB-PROG    | PIC X(8)            | El nombre del programa de manejador de datos ficticio.                                                                                 |
| relleno        | PIC X               |                                                                                                                                        |
| DSVSAM-PROG    | PIC X(8)            | El nombre del programa de manejador de datos VSAM                                                                                      |
| relleno        | PIC X               |                                                                                                                                        |
| ODSTUB-PROG    | PIC X(8)            | El nombre del módulo de asignador de pedidos ficticio.                                                                                 |
| relleno        | PIC X               |                                                                                                                                        |

Tabla 27. Registro de información general (continuación)

| Nombre      | Tipo de datos COBOL | Descripción                                                             |
|-------------|---------------------|-------------------------------------------------------------------------|
| ODWEBS-PROG | PIC X(8)            | El nombre del programa asignador de pedidos del servicio web de salida. |
| relleno     | PIC X               |                                                                         |
| STKMAN-PROG | PIC X(8)            | El nombre del programa de gestor de stock.                              |
| relleno     | PIC X(10)           |                                                                         |

Tabla 28. Registro URL de salida

| Nombre       | Tipo de datos COBOL | Descripción                                                                    |
|--------------|---------------------|--------------------------------------------------------------------------------|
| URL-KEY      | PIC X(9)            | Campo de clave para el registro de información general que contiene OUTBNDURL. |
| relleno      | PIC X               |                                                                                |
| OUTBOUND-URL | PIC X(255)          | URL de salida para la solicitud de servicio web del asignador de pedidos.      |

Tabla 29. Registro de información del archivo de catálogo

| Nombre            | Tipo de datos COBOL | Descripción                                                                    |
|-------------------|---------------------|--------------------------------------------------------------------------------|
| URL-FILE-KEY      | PIC X(9)            | Campo de clave para el registro de información general que contiene VSAM-NAME. |
| relleno           | PIC X               |                                                                                |
| CATALOG-FILE-NAME | PIC X(8)            | Nombre del recurso CICS FILE utilizado para el archivo de catálogo.            |

Tabla 30. Registro de información de servidor

| Nombre            | Tipo de datos COBOL | Descripción                                                                                                                                                |
|-------------------|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| WS-SERVER-KEY     | PIC X(9)            | Campo de clave para el registro de información de servidor que contiene WS-SERVER.                                                                         |
| relleno           | PIC X               |                                                                                                                                                            |
| CATALOG-FILE-NAME | PIC X(8)            | Sólo para el cliente de servicio web CICS, la dirección IP y el puerto del sistema en el que se va a desplegar la aplicación de ejemplo como servicio web. |

## Ejemplos de JSON

Utilice estos ejemplos para comprender las solicitudes JSON.

## Ejemplo de solicitud de HTTP GET mediante una cadena de consulta

A continuación, se presenta un ejemplo de una solicitud HTTP GET mediante una cadena de consulta.

```
GET
/genapp/customers?name=Joe%20Bloggs/ ❶
Host: www.example.com
```

Donde ❶ es la cadena de consulta.

## Solicitud de ejemplo HTTP con un cuerpo JSON

A continuación, se muestra un ejemplo de solicitud HTTP con un cuerpo JSON.

```
POST /genapp/customers/
Host: www.example.com
Content-Type: application/json
Content-Length: nn ❶

{
 "customers":
 {
 "firstName": "Joe",
 "lastName": "Bloggs",
 "fullAddress":
 {
 "streetAddress": "21 2nd Street",
 "city": "New York",
 "state": "NY",
 "postalCode": 10021
 }
 }
}
```

Donde Content-Length: nn ❶ es la longitud de la solicitud.

La correlación de estructura de lenguaje COBOL para este ejemplo sería la siguiente:

```
01 CUSTOMERS.
 03 firstname pic x(8).
 03 lastname pic x(8).
 03 fulladdress.
 05 streetaddress pic x(20).
 05 city pic x(20).
 05 state pic xx.
 05 postalcode pic 9(5).
```



---

## Notices

This information was developed for products and services offered in the U.S.A. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property rights may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing  
IBM Corporation  
North Castle Drive, MD-NC119  
Armonk, NY 10504-1785  
United States of America*

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

*Intellectual Property Licensing  
Legal and Intellectual Property Law  
IBM Japan Ltd.  
19-21, Nihonbashi-Hakozakicho, Chuo-ku  
Tokyo 103-8510, Japan*

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who want to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact

*IBM Director of Licensing  
IBM Corporation  
North Castle Drive, MD-NC119  
Armonk, NY 10504-1785  
US*

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Programming License Agreement, or any equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

#### **COPYRIGHT LICENSE:**

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

### **Programming interface information**

CICS supplies some documentation that can be considered to be Programming Interfaces, and some documentation that cannot be considered to be a Programming Interface.

Programming Interfaces that allow the customer to write programs to obtain the services of CICS Transaction Server for z/OS, Version 5 Release 4 are included in the following sections of the online product documentation:

- Developing applications
- Developing system programs
- RACF security overview
- Developing for external interfaces
- Reference: application development
- Reference: system programming
- Reference: connectivity

Information that is NOT intended to be used as a Programming Interface of CICS Transaction Server for z/OS, Version 5 Release 4 , but that might be misconstrued as Programming Interfaces, is included in the following sections of the online product documentation:

- Troubleshooting and support
- Reference: diagnostics

If you access the CICS documentation in manuals in PDF format, Programming Interfaces that allow the customer to write programs to obtain the services of CICS Transaction Server for z/OS, Version 5 Release 4 are included in the following manuals:

- Application Programming Guide and Application Programming Reference
- Business Transaction Services
- Customization Guide
- C++ OO Class Libraries
- Debugging Tools Interfaces Reference
- Distributed Transaction Programming Guide
- External Interfaces Guide
- Front End Programming Interface Guide
- IMS Database Control Guide
- Installation Guide
- Security Guide
- Supplied Transactions
- CICSplex SM Managing Workloads
- CICSplex SM Managing Resource Usage
- CICSplex SM Application Programming Guide and Application Programming Reference
- Java Applications in CICS

If you access the CICS documentation in manuals in PDF format, information that is NOT intended to be used as a Programming Interface of CICS Transaction Server for z/OS, Version 5 Release 4 , but that might be misconstrued as Programming Interfaces, is included in the following manuals:

- Data Areas
- Diagnosis Reference
- Problem Determination Guide
- CICSplex SM Problem Determination Guide

## Trademarks

IBM, the IBM logo, and [ibm.com](http://ibm.com) are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at Copyright and trademark information at [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

## Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

### Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

### Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

### Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

**Rights** Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.



IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

## **IBM online privacy statement**

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below:

### **For the CICSplex SM Web User Interface (main interface):**

Depending upon the configurations deployed, this Software Offering may use session and persistent cookies that collect each user's user name and other personally identifiable information for purposes of session management, authentication, enhanced user usability, or other usage tracking or functional purposes. These cookies cannot be disabled.

### **For the CICSplex SM Web User Interface (data interface):**

Depending upon the configurations deployed, this Software Offering may use session cookies that collect each user's user name and other personally identifiable information for purposes of session management, authentication, or other usage tracking or functional purposes. These cookies cannot be disabled.

### **For the CICSplex SM Web User Interface ("hello world" page):**

Depending upon the configurations deployed, this Software Offering may use session cookies that collect no personally identifiable information. These cookies cannot be disabled.

### **For CICS Explorer:**

Depending upon the configurations deployed, this Software Offering may use session and persistent preferences that collect each user's user name and password, for purposes of session management, authentication, and single sign-on configuration. These preferences cannot be disabled, although storing a user's password on disk in encrypted form can only be enabled by the user's explicit action to check a check box during sign-on.

If the configurations deployed for this Software Offering provide you, as customer, the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM Privacy Policy and IBM Online Privacy Statement, the section entitled “Cookies, Web Beacons and Other Technologies” and the IBM Software Products and Software-as-a-Service Privacy Statement.

# Índice

## Caracteres Especiales

- <addressing>
  - elemento de configuración de interconexiones 99
- <apphandler\_class>
  - elemento de configuración de interconexiones 94
- <apphandler>
  - elemento de configuración de interconexiones 95
- <auth\_token\_type>
  - elemento de configuración de interconexiones 131
- <authentication>
  - elemento de configuración de interconexiones 124
- <cics\_json\_handler\_java>
  - elemento de configuración de interconexiones 100
- <cics\_mtom\_handler>
  - elemento de configuración de interconexiones 116
- <cics\_soap\_1.1\_handler>
  - elemento de configuración de interconexiones 100
- <cics\_soap\_1.1\_handler\_java>
  - elemento de configuración de interconexiones 102
- <cics\_soap\_1.2\_handler>
  - elemento de configuración de interconexiones 105
- <cics\_soap\_1.2\_handler\_java>
  - elemento de configuración de interconexiones 107
- <default\_http\_transport\_handler\_list>
  - elemento de configuración de interconexiones 109
- <default\_mq\_transport\_handler\_list>
  - elemento de configuración de interconexiones 110
- <default\_target>
  - elemento de configuración de interconexiones 115
- <default\_transport\_handler\_list>
  - elemento de configuración de interconexiones 110
- <dfhmtom\_configuration>
  - elemento de configuración de interconexiones 117
- <dfhwsse\_configuration>
  - elemento de configuración de interconexiones 122
- <encrypt\_body>
  - elemento de configuración de interconexiones 133
- <handler>
  - elemento de configuración de interconexiones 111

- <jvmserver>
  - elemento de configuración de interconexiones 112
- <mime\_options>
  - elemento de configuración de interconexiones 121
- <mtom>
  - elemento de configuración de interconexiones 116
- <mtom\_options>
  - elemento de configuración de interconexiones 118
- <named\_transport\_entry>
  - elemento de configuración de interconexiones 95
- <namespace>
  - elemento de configuración de interconexiones 99
- <provider\_pipeline>
  - elemento de configuración de interconexiones 95
- <provider\_pipeline\_json>
  - elemento de configuración de interconexiones 96
- <repository>
  - elemento de configuración de interconexiones 112
- <requester\_pipeline>
  - elemento de configuración de interconexiones 98
- <service>
  - elemento de configuración de interconexiones 113
- <service\_handler\_list>
  - elemento de configuración de interconexiones 113
- <service\_parameter\_list>
  - elemento de configuración de interconexiones 114
- <sign\_body>
  - elemento de configuración de interconexiones 133
- <sts\_authentication>
  - elemento de configuración de interconexiones 129
- <sts\_endpoint>
  - elemento de configuración de interconexiones 132
- <terminal\_handler>
  - elemento de configuración de interconexiones 97
- <transport>
  - elemento de configuración de interconexiones 115
- <transport\_handler\_list>
  - elemento de configuración de interconexiones 98
- <wsse\_handler>
  - elemento de configuración de interconexiones 122

- <xop\_options>
  - elemento de configuración de interconexiones 119

## A

- addressing
  - elemento de configuración de interconexiones 99
- algoritmo 612, 614
- análisis de XML 400
- aplicación
  - transformar en XML 291
- aplicación de proveedor de servicios 274
  - creación a partir de una estructura de datos 275, 579
  - utilización de transacciones atómicas 197
- aplicación de solicitante de servicio
  - utilización de transacciones atómicas 198
- apphandler
  - elemento de configuración de interconexiones 95
- archivo adjunto binario
  - configuración de interconexiones 116
- archivo de configuración, interconexión 81
- archivo de configuración de interconexión 81
- archivo de enlace
  - enlace XML 404
- asistente, JSON 236, 411
- asistente, servicios web 489
- asistente, XML 291
- asistente de JSON 236, 411, 640, 643
- asistente de servicios web 489
  - creación de una aplicación de proveedor de servicios 275, 579
- asistente XML 291
  - DFHLS2SC 404
- autenticación
  - elemento de configuración de interconexiones 124
- auth\_token\_type
  - elemento de configuración de interconexiones 131
- Axis2 57

## C

- C y C++
  - correlación con esquema JSON 451, 454, 545, 548
  - correlación con esquema XML 329, 332
- cabecera, SOAP 20
- cics\_json\_handler\_java
  - elemento de configuración de interconexiones 100

- cics\_mtom\_handler
  - elemento de configuración de interconexiones 116
- cics\_soap\_1.1\_handler
  - elemento de configuración de interconexiones 100
- cics\_soap\_1.1\_handler.java
  - elemento de configuración de interconexiones 102
- cics\_soap\_1.2\_handler
  - elemento de configuración de interconexiones 105
- cics\_soap\_1.2\_handler.java
  - elemento de configuración de interconexiones 107
- cliente de confianza
  - interfaz 611
  - invocación 624
- COBOL
  - contenido de repetición variable 362, 392
  - correlación con esquema JSON 436, 530
  - Correlación con esquema JSON 444, 538
  - correlación con esquema XML 313, 322
- código de retorno 643
- configuración 76
- configuración de interconexión 620
- configuración de interconexiones
  - MTOM/XOP 116
  - Seguridad de servicios web 121
- configuración de RACF 615
- consulta XML 399
- contenedor
  - contenedor de contexto
    - DFH-HANDLERPLIST 162
    - DFH-SERVICEPLIST 162
    - DFHWS-APPHANDLER 162, 163, 165
    - DFHWS-DATA 163
    - DFHWS-FAULT 164
    - DFHWS-PIPELINE 165
    - DFHWS-SOAPLEVEL 166
    - DFHWS-STSREASON 178
    - DFHWS-TRANID 166
    - DFHWS-URI 166
    - DFHWS-USERID 170
    - DFHWS-WEBSERVICE 171
  - contenedor de control
    - DFHERROR 152
    - DFHFUNCTION 154
    - DFHHTTPSTATUS 156
    - DFHMEDIATYPE 157
    - DFHNORESPONSE 157
    - DFHREQUEST 157
    - DFHRESPONSE 158
    - DFHWS-CCSID 158
  - DFH-EXIT-HEADER1 162
  - DFH-HANDLERPLIST 162
  - DFH-SERVICEPLIST 162
  - DFHERROR 152
  - DFHFUNCTION 154
  - DFHHTTPSTATUS 156
  - DFHMEDIATYPE 157
  - DFHNORESPONSE 157
- contenedor (*continuación*)
  - DFHREQUEST 157
  - DFHRESPONSE 158
  - DFHWS-APPHANDLER 162, 163, 165
  - DFHWS-CCSID 158
  - DFHWS-CID-DOMAIN 171
  - DFHWS-DATA 163
  - DFHWS-FAULT 164
  - DFHWS-IDTOKEN 176
  - DFHWS-LOCATION 164
  - DFHWS-MEP 164
  - DFHWS-MTOM-IN 171
  - DFHWS-MTOM-OUT 172
  - DFHWS-PIPELINE 165
  - DFHWS-RESPWAIT 165
  - DFHWS-RESTOKEN 177
  - DFHWS-SERVICEURI 177
  - DFHWS-SOAPLEVEL 166
  - DFHWS-STSACTION 177
  - DFHWS-STSFAULT 178
  - DFHWS-STSREASON 178
  - DFHWS-STSURI 178
  - DFHWS-TOKENTYPE 178
  - DFHWS-TRANID 166
  - DFHWS-URI 166
  - DFHWS-USERID 170
  - DFHWS-WEBSERVICE 171
  - DFHWS-XOP-IN 174
  - DFHWS-XOP-OUT 173, 174
- contenedor de contexto
  - DFH-EXIT-HEADER1 162
  - DFHWS-CID-DOMAIN 171
  - DFHWS-IDTOKEN 176
  - DFHWS-LOCATION 164
  - DFHWS-MEP 164
  - DFHWS-MTOM-IN 171
  - DFHWS-MTOM-OUT 172
  - DFHWS-RESPWAIT 165
  - DFHWS-RESTOKEN 177
  - DFHWS-SERVICEURI 177
  - DFHWS-STSACTION 177
  - DFHWS-STSFAULT 178
  - DFHWS-STSURI 178
  - DFHWS-TOKENTYPE 178
  - DFHWS-WSDL-CTX 173
  - DFHWS-XOP-IN 174
  - DFHWS-XOP-OUT 174
- Contenedor DFH-EXIT-HEADER1 162
- Contenedor DFH-HANDLERPLIST 162
- contenedor DFH-SERVICEPLIST 162
- contenedor DFHERROR 152
- contenedor DFHFUNCTION 154
- Contenedor DFHHTTPSTATUS 156
- Contenedor DFHMEDIATYPE 157
- contenedor DFHNORESPONSE 157
- contenedor DFHREQUEST 157
- contenedor DFHRESPONSE 158
- contenedor DFHSAML-
  - AnnnVmmm 179
- contenedor DFHSAML-ASSQNAME 179
- contenedor DFHSAML-ATTRAnnn 179
- contenedor DFHSAML-ATTRFnnn 179
- contenedor DFHSAML-ATTRNnnn 179
- contenedor DFHSAML-ATTRSnnn 180
- contenedor DFHSAML-ATTRYnnn 180
- contenedor DFHSAML-AUDNRnnn 180

- contenedor DFHSAML-
- AUTHMETH 180
- contenedor DFHSAML-CERTIDN 180
- contenedor DFHSAML-CERTSDN 180
- contenedor DFHSAML-CERTSNUM 180
- contenedor DFHSAML-
- CONFMETH 180
- contenedor DFHSAML-COUNTS 181
- contenedor DFHSAML-FLAGS 181
- contenedor DFHSAML-ISSUER 181
- contenedor DFHSAML-NAMID 181
- contenedor DFHSAML-NAMIDF 181
- contenedor DFHSAML-NAMIDQ 181
- contenedor DFHSAML-NAMIDSP 181
- contenedor DFHSAML-NAMIDSPQ 181
- contenedor DFHSAML-OUTTOKEN 181
- contenedor DFHSAML-PROXYnnn 182
- contenedor DFHSAML-RESPONSE 182
- contenedor DFHSAML-SAMLID 182
- contenedor DFHSAML-SUBJADDR 182
- contenedor DFHSAML-SUBJDNS 182
- contenedor DFHSAML-TIMES 182
- contenedor DFHWS-
- APPHANDLER 162, 163, 165
- Contenedor DFHWS-CCSID 158
- Contenedor DFHWS-CID-DOMAIN 171
- contenedor DFHWS-DATA 163
- contenedor DFHWS-FAULT 164
- Contenedor DFHWS-IDTOKEN 176
- Contenedor DFHWS-LOCATION 164
- Contenedor DFHWS-MEP 164
- Contenedor DFHWS-MTOM-IN 171
- Contenedor DFHWS-MTOM-OUT 172
- Contenedor DFHWS-PIPELINE 165
- Contenedor DFHWS-RESPWAIT 165
- Contenedor DFHWS-RESTOKEN 177
- Contenedor DFHWS-SERVICEURI 177
- contenedor DFHWS-SOAPLEVEL 166
- Contenedor DFHWS-STSACTION 177
- Contenedor DFHWS-STSFAULT 178
- Contenedor DFHWS-STSREASON 178
- Contenedor DFHWS-STSURI 178
- Contenedor DFHWS-TOKENTYPE 178
- contenedor DFHWS-TRANID 166
- Contenedor DFHWS-URI 166
- contenedor DFHWS-USERID 170
- contenedor DFHWS-WEBSERVICE 171
- Contenedor DFHWS-WSDL-CTX 173
- Contenedor DFHWS-XOP-IN 174
- Contenedor DFHWS-XOP-OUT 174
- contenedores
- descripción de canal 277, 582
- DFHSAML-AnnnVmmm 179
- DFHSAML-ASSQNAME 179
- DFHSAML-ATTRAnnn 179
- DFHSAML-ATTRFnnn 179
- DFHSAML-ATTRNnnn 179
- DFHSAML-ATTRSnnn 180
- DFHSAML-ATTRYnnn 180
- DFHSAML-AUDNRnnn 180
- DFHSAML-AUTHMETH 180
- DFHSAML-CERTIDN 180
- DFHSAML-CERTSDN 180
- DFHSAML-CERTSNUM 180
- DFHSAML-CONFMETH 180
- DFHSAML-COUNTS 181
- DFHSAML-FLAGS 181

- contenedores (*continuación*)
  - DFHSAML-ISSUER 181
  - DFHSAML-NAMID 181
  - DFHSAML-NAMIDF 181
  - DFHSAML-NAMIDQ 181
  - DFHSAML-NAMIDSP 181
  - DFHSAML-NAMIDSPQ 181
  - DFHSAML-OUTTOKEN 181
  - DFHSAML-PROXYnnn 182
  - DFHSAML-RESPONSE 182
  - DFHSAML-SAMLID 182
  - DFHSAML-SUBJADDR 182
  - DFHSAML-SUBJDNS 182
  - DFHSAML-TIMES 182
  - SAML 178
- contenedores de contexto 161
- contenedores de control 152
- contenedores de seguridad 176
  - DFHSAML-AnnnVmmm 179
  - DFHSAML-ASSQNAME 179
  - DFHSAML-ATTRAnnn 179
  - DFHSAML-ATTRFnnn 179
  - DFHSAML-ATTRNnnn 179
  - DFHSAML-ATTRSnnn 180
  - DFHSAML-ATTRYnnn 180
  - DFHSAML-AUDNRnnn 180
  - DFHSAML-AUTHMETH 180
  - DFHSAML-CERTIDN 180
  - DFHSAML-CERTSDN 180
  - DFHSAML-CERTSNUM 180
  - DFHSAML-CONFMETH 180
  - DFHSAML-COUNTS 181
  - DFHSAML-FLAGS 181
  - DFHSAML-ISSUER 181
  - DFHSAML-NAMID 181
  - DFHSAML-NAMIDF 181
  - DFHSAML-NAMIDQ 181
  - DFHSAML-NAMIDSP 181
  - DFHSAML-NAMIDSPQ 181
  - DFHSAML-OUTTOKEN 181
  - DFHSAML-PROXYnnn 182
  - DFHSAML-RESPONSE 182
  - DFHSAML-SAMLID 182
  - DFHSAML-SUBJADDR 182
  - DFHSAML-SUBJDNS 182
  - DFHSAML-TIMES 182
- contenedores de usuario 183
- contenido que se repite 362, 392
- correlación con C y C++ 329, 332, 451, 454, 545, 548
- correlación con COBOL 313, 322, 436, 444, 530, 538
- correlación con PL/I 337, 342, 460, 466, 554, 560
- correlación de esquemas XML con datos de aplicación 406
- correlación de estructuras de lenguaje a XML 404
- Creación de una aplicación de solicitante de servicio 487
- cuerpo, SOAP 20

## D

- default\_http\_transport\_handler\_list
  - elemento de configuración de interconexiones 109, 112

- default\_mq\_transport\_handler\_list
  - elemento de configuración de interconexiones 110
- default\_target
  - elemento de configuración de interconexiones 115
- default\_transport\_handler\_list
  - elemento de configuración de interconexiones 110
- definición de interconexión
  - solicitante de servicio 92
- descripción de canal 277, 582
- descubrimiento de servicios web 45
- determinación de problemas 639
- DFHJS2LS
  - procedimiento catalogado 248, 261, 419
- DFHLS2JS
  - procedimiento catalogado 236, 412
- DFHLS2SC
  - procedimiento catalogado 292
- DFHLS2SC, programa de asistente XML de CICS 404
- DFHLS2WS
  - procedimiento catalogado 490
- dfhmtom\_configuration
  - elemento de configuración de interconexiones 117
- DFHSC2LS
  - procedimiento catalogado 300
- DFHWS2LS
  - procedimiento catalogado 506
- dfhwsse\_configuration
  - elemento de configuración de interconexiones 122
- diagnóstico de problemas
  - proveedor de servicios 632
  - solicitante de servicio 633
- direccionamiento dinámico
  - en un controlador de terminal 150
  - en un proveedor de servicios 150
- documento WSDL
  - espacio en blanco 370, 395, 571
  - longitud variable 370, 395, 571

## E

- ejemplo 645
- ejemplo de catálogoejemplo de catálogo 645
- Ejemplos de JSON 691
- elemento de configuración de interconexiones
  - <addressing> 99
  - <apphandler> 95
  - <auth\_token\_type> 131
  - <authentication> 124
  - <cics\_json\_handler\_java> 100
  - <cics\_mtom\_handler> 116
  - <cics\_soap\_1.1\_handler> 100
  - <cics\_soap\_1.1\_handler\_java> 102
  - <cics\_soap\_1.2\_handler> 105
  - <cics\_soap\_1.2\_handler\_java> 107
  - <default\_http\_transport\_handler\_list> 109
  - <default\_mq\_transport\_handler\_list> 110

- elemento de configuración de interconexiones (*continuación*)
  - <default\_transport\_handler\_list> 110
  - <dfhmtom\_configuration> 117
  - <dfhwsse\_configuration> 122
  - <encrypt\_body> 133
  - <handler> 111
  - <jvmserver> 112
  - <mime\_options> 121
  - <mtom> 116
  - <mtom\_options> 118
  - <named\_transport\_entry> 95
  - <namespace> 99
  - <provider\_pipeline> 95
  - <provider\_pipeline.json> 96
  - <repository> 112
  - <requester\_pipeline> 98
  - <service> 113
  - <service\_handler\_list> 113
  - <sign\_body> 133
  - <sts\_authentication> 129
  - <sts\_endpoint> 132
  - <terminal\_handler> 97
  - <transport> 115
  - <transport\_handler\_list> 98
  - <wsse\_handler> 122
  - <xop\_options> 119
- encrypt\_body
  - elemento de configuración de interconexiones 133
- enlace XML 404
- EPR predeterminada 220
- WSDL 1.1 220
- error 643
- error, SOAP 20
- errores de servicio web 632, 633
- errores SOAP 585
- espacio de nombres
  - elemento de configuración de interconexiones 99
- esquema
  - descripción de canal 277, 582
- esquema JSON 274, 436, 444, 451, 454, 460, 466, 530, 538, 545, 548, 554, 560
  - conversión a JSON 412
- esquema XML 322, 329, 332
  - conversión a WSDL 292
  - espacio en blanco 370, 395, 571
  - longitud variable 370, 395, 571
- Esquema XML 313, 337, 342
- estructura de lenguaje
  - conversión a JSON 236, 412
  - conversión a WSDL 292, 490
- estructura de lenguaje de alto nivel
  - conversión a JSON 236
  - conversión a WSDL 490

## F

- feed Atom
  - enlace XML
    - creación 404

## G

Generación de correlaciones a partir de estructuras de lenguaje 480  
Generación de correlaciones a partir de un esquema JSON 482  
gestión de carga de trabajo  
    en un controlador de terminal 150  
    en un proveedor de servicios 150  
GLUEs 189

## H

handler  
    elemento de configuración de interconexiones 111

## I

instalación 639  
instalar 639  
interconexiones SOAP 57  
interconexiones SOAP basadas en Java 216  
interfaz de canal 277, 582  
invocación del cliente de confianza 624

## J

Java 57  
    servicio web de llamada 598  
JSON 76, 169, 170, 639, 640, 641, 642, 643, 691  
    conversión a estructura de lenguaje 419

## L

limitaciones de tiempo de ejecución 186  
limitaciones en tiempo de ejecución 186  
línea de mandatos 643  
lista de parámetros de servicio  
    <service\_parameter\_list> 114

## M

mandato EXEC CICS SOAPFAULT CREATE 585  
manejador de aplicación  
    elemento de configuración de interconexiones 94  
manejador de mensajes  
    invocación del cliente de confianza 624  
    que no es de terminal 140  
    que no son de terminal 142, 143  
manejador de mensajes que no es de terminal 140  
manejador de mensajes que no son de terminal 142, 143  
manejador de seguridad  
    cómo escribir su propio 623  
manejador de seguridad  
    personalizado 623

Matrices de variables 284, 349, 365, 373, 378, 473, 567  
maxOccurs  
    en esquema JSON 284, 373, 473  
    en esquema XML 349, 365, 378, 567  
Mensaje MIME  
    configuración de interconexiones 116  
mensaje permanente 51  
mensaje SOAP  
    ejemplo 20  
    estructura 20  
Mensaje SOAP  
    cifrado 613  
    firma 612  
mensajes SOAP  
    esquema XML  
        validación de mensaje SOAP 599  
        validación en el esquema XML 599  
MEP 13  
mime\_options  
    elemento de configuración de interconexiones 121  
minOccurs  
    en esquema JSON 284, 373, 473  
    en esquema XML 349, 365, 378, 567  
modalidad de compatibilidad 203  
modalidad directa 203  
mtom  
    elemento de configuración de interconexiones 116  
mtom\_options  
    elemento de configuración de interconexiones 118  
MTOM/XOP  
    configuración de interconexiones 116

## N

named\_transport\_entry  
    elemento de configuración de interconexiones 95

## P

patrón de intercambio de mensajes (MEP) 13  
personalización del proceso de interconexión 189  
PL/I  
    correlación con esquema JSON 460, 466, 554, 560  
    correlación con esquema XML 337, 342  
proceso de interconexión  
    personalización 189  
    sustitución del URI 191  
programa de utilidad  
    asistente de JSON 236, 411  
    asistente de servicios web 489  
    asistente XML 291  
programa de utilidad por lotes 236  
    asistente de JSON 411  
    asistente de servicios web 489  
    asistente XML 291  
proveedor de servicios  
    diagnóstico de problemas 632

provider\_pipeline  
    elemento de configuración de interconexiones 95

## R

referencia de punto final  
    predeterminada 220  
referencia de punto final (EPR) 212  
requester\_pipeline  
    elemento de configuración de interconexiones 98  
    elemento de definición de interconexión 92  
resolución de problemas 639, 643

## S

salidas de usuario globales 189  
SAML  
    contenedores 178  
script dinámico 639  
Seguridad de servicios web  
    configuración de interconexiones 121  
Seguridad de servicios web (WSS) 605, 615, 620  
seguridad para servicios web 605  
service  
    elemento de configuración de interconexiones 113  
service\_handler\_list  
    elemento de configuración de interconexiones 113  
service\_parameter\_list  
    lista de parámetros de servicio 114  
Servicio de señales de seguridad  
    interfaz de cliente de confianza 611  
servicio web de llamada desde Java 598  
servicios web de CICS  
    Unicode 397, 478, 574  
    UTF-16 397, 478, 574  
servidor de JVM 57, 598  
sign\_body  
    elemento de configuración de interconexiones 133  
SOAP  
    cabecera 20  
    cuerpo 20  
    descripción general 15  
    descripción general de SOAP 15  
    error 20  
    sobre 20  
sobre, SOAP 20  
solicitante de servicio  
    definición de interconexión 92  
    diagnóstico de problemas 633  
soporte de mensajes permanentes 52  
sts\_authentication  
    elemento de configuración de interconexiones 129  
sts\_endpoint  
    elemento de configuración de interconexiones 132  
sustitución del URI 191

## T

- terminal\_handler
  - elemento de configuración de interconexiones 97
- tipos de datos xsd:any 402
- transacción atómica 193, 200
  - configuración de CICS 195
  - configuración del proveedor de servicios 197
  - configuración del solicitante de servicio 198
  - estados 201
  - servicios de registro 193
- TRANSFORM DATATOXML 407
- TRANSFORM XMLTODATA 408
- transformación de datos binarios en XML 407
- Transformación de datos de aplicación en JSON 483
- Transformación de JSON en datos de aplicación 485
- Transformación de JSON mediante la interfaz enlazable 410
- transformación de XML 400
- transformación de XML en datos binarios 408
- transport\_handler\_list
  - elemento de configuración de interconexiones 98
- transporte
  - elemento de configuración de interconexiones 115

## U

- URI
  - para el transporte WebSphere MQ 49

## V

- validación de mensajes SOAP 599
- Vía de acceso de mensaje SOAP 8

## W

- Web Services Addressing
  - <wsa:Action> 221
  - acciones explícitas 219, 221
  - acciones predeterminadas 219, 222, 223
  - configuración de interconexión de proveedor 217
  - configuración de interconexión de solicitante 216
  - DFHWS-URI 212
  - DFHWSADH 216, 217
  - EPR 212
  - EPR predeterminada 219, 220
  - especificación 211
  - manejador de direccionamiento 216, 217
  - MAP 212
  - parámetros 522
  - solicitante de servicio 219
  - soporte 211

- Web Services Addressing (*continuación*)
  - WSADDR-EPR-ANY 522
  - WSDL 1.1 222
  - WSDL 2.0 223
- WS-Addressing
  - <wsa:Action> 221
  - acciones explícitas 221
  - acciones predeterminadas 222, 223
  - configuración de interconexión de proveedor 217
  - configuración de interconexión de solicitante 216
  - DFHWS-URI 212
  - DFHWSADH 216, 217
  - EPR 212
  - EPR predeterminada 220
  - especificación 211
  - manejador de direccionamiento 216, 217
  - MAP 212
  - parámetros 522
  - WSADDR-EPR-ANY 522
  - WSDL 1.1 222
  - WSDL 2.0 223
- WS-AT 193
- WSDL
  - consulta 45
  - conversión a estructura de lenguaje 248, 261, 300, 506
  - Web Services Addressing 219
  - y la estructura de datos de la aplicación 11
- WSDL 1.1
  - EPR predeterminada 220
- wsse\_handler
  - elemento de configuración de interconexiones 122

## X

- XML
  - análisis 400
  - asistente 291
  - consulta 399
  - tipos de datos 400
  - tipos de datos xsd:any 402
  - transformación 400
  - transformar en datos 291
- xop\_options
  - elemento de configuración de interconexiones 119

## Z

- z/OS Connect 76, 79
  - configuración 72, 74
  - descripción general 36
  - Seguridad 626
- zAAP 57
- zosConnectService 76







