

CICS Transaction Server for z/OS



Usando serviços da web com CICS

Versão 5 Release 4

CICS Transaction Server for z/OS



Usando serviços da web com CICS

Versão 5 Release 4

Nota

Antes de utilizar estas informações e o produto que elas suportam, leia as informações em “Avisos” na página 691.

Índice

Sobre este PDF v

Capítulo 1. CICS e Serviços da Web. . . 1

CICS e serviços da web SOAP	4
Nós SOAP	5
Mensagens SOAP e a Estrutura de Dados do Aplicativo	9
WSDL e a Estrutura de Dados do Aplicativo	11
WSDL e Message Exchange Patterns	13
O Arquivo de Ligação de Serviço da Web	15
Normas Externas	15
Arquitetura SOAP e formato da mensagem.	15
Planejando para usar serviços da web SOAP	26
Serviços da web do CICS e JSON	27
Conceitos de serviços da web JSON	28
Conceitos de serviços da web JSON RESTful	30
Planejando usar serviços da web JSON	32
CICS e z/OS Connect	35

Capítulo 2. Configurando Serviços da Web no CICS 39

Configurando seu Sistema CICS para Serviços da Web	39
Recursos do CICS para serviços da web	39
Configurando o CICS para Usar o Transporte do WebSphere MQ	43
Interoperabilidade Entre o Assistente de Serviços da Web e o WSRR	51
Criando a Infraestrutura de Serviços da Web	52
A Infraestrutura de Serviços da Web	52
Criando a infraestrutura do CICS para um provedor de serviços SOAP	62
Criando a infraestrutura do CICS para um solicitante de serviço SOAP	65
Criando a infraestrutura do CICS para um provedor de serviços JSON	67
Criando a infraestrutura do CICS para um provedor de serviços JSON não Java	68
Configurando o z/OS Connect for CICS.	69
Arquivos de configuração de pipeline	79
Manipuladores de Aplicativo	132
Manipuladores de mensagens	134
Os Manipuladores de Mensagem SOAP	143
Contêineres usados no pipeline	148
Processamento de Tempo de Execução para Serviços da Web	181
Suporte para Transações de Serviços da Web	190
Suporte para Otimização de MTOM/XOP de Dados Binários	199
Suporte para Web Services Addressing	209
Suporte para SAML	231

Capítulo 3. Desenvolvendo serviços da web 233

Criando um serviço da web JSON	233
O assistente JSON do CICS.	233
Criando um aplicativo do provedor de serviços JSON usando o assistente JSON	272
Restrições de serviço da web JSON	287
Mapeando e transformando dados do aplicativo e XML	289
O assistente XML do CICS	290
Gerando mapeamentos a partir de estruturas de linguagem	402
Gerando mapeamentos a partir de um esquema XML	404
Transformando dados do aplicativo em XML	405
Transformando XML em dados do aplicativo	407
Mapeando e transformando dados do aplicativo e JSON	408
O assistente JSON do CICS.	409
Gerando mapeamentos a partir da estrutura de linguagem	478
Gerando mapeamentos a partir de um esquema JSON	479
Transformando dados do aplicativo em JSON vinculando-se ao DFHJSON	481
Transformando dados do aplicativo em JSON usando o comando de API TRANSFORM DATATOJSON	482
Transformando JSON em dados do aplicativo vinculando-se ao DFHJSON	483
Transformando JSON em dados do aplicativo usando o comando de API TRANSFORM JSONTODATA	484
Criando um aplicativo cliente de serviço da web JSON	485
Criando um serviço da web SOAP	486
O assistente de serviços da web do CICS	487
Criando um Provedor de Serviço da Web Usando o Assistente de Serviços da Web.	574
Criando um aplicativo do provedor de serviços a partir de uma descrição de serviços da web	574
Criando um Aplicativo do Provedor de Serviços a Partir de uma Estrutura de Dados.	577
Criando um Documento de Descrição de Canal Customizando Documentos da Descrição do Serviço da Web Gerado	582
Enviando uma Falha de SOAP	584
Criando um solicitante de serviço da web usando o assistente de serviços da web	586
Criando um Serviço da Web Usando o Conjunto de Ferramentas	588
Criando Seus Próprios Aplicativos de Serviço da Web que Reconhecem XML.	589
Criando um Aplicativo do Provedor de Serviços que Reconhece XML	589

Criando um Aplicativo do Solicitante de Serviço que Reconhece XML	591
Usando Java com serviços da web	593
Implementando um serviço da web no modo de provedor Java em um servidor JVM Axis2.	593
Criando um serviço da web Java que gera e analisa XML.	595
Criando um serviço da web Java que possui uma interface COBOL	596
Implementando um serviço da Web JAX-WS no modo do solicitante	596
Implementando um serviço da web no modo de provedor Java em um servidor Liberty JVM	597
Validando Mensagens SOAP	597
Manipulando dados fornecidos pelo aplicativo inválidos e não inicializados	599
Exemplo 1: tolerância de campos decimais	600

Capítulo 4. Suporte para proteger

serviços da web 603

Pré-requisitos para Web Services Security	604
Planejamento para proteger serviços da web SOAP	604
Opções para Proteger Mensagens SOAP	605
Autenticação utilizando um Security Token Service	608
A Interface do Cliente de Confiança	609
Assinatura de Mensagens SOAP	610
Algoritmos de Assinatura	610
Exemplo de uma Mensagem SOAP Assinada	611
Suporte CICS para Mensagens SOAP	
Criptografadas	612
Algoritmos de Criptografia	612
Exemplo de uma Mensagem SOAP	
Criptografada	613
Configurando o RACF para Web Services Security	614
Configurando Serviços da Web no Modo de	
Provedor para Propagação de Identidade	616
Configurando o Pipeline para Web Services	
Security	618
Gravando um Manipulador de Segurança	
Customizada	622
Chamando o Cliente de Confiança a Partir de um	
Manipulador de Mensagem	623
Segurança para o z/OS Connect	624
Configurando permissões para Serviços e APIs	
do z/OS Connect	625

Capítulo 5. Resolvendo Problemas de Serviços da Web 627

Resolução de problemas de serviços da web SOAP	627
Diagnosticando erros de implementação	627
Diagnosticando Erros de Tempo de Execução do	
Provedor de Serviços.	630
Diagnosticando Erros de Tempo de Execução do	
Solicitante de Serviço.	631
Diagnosticando Erros de MTOM/XOP	633
Diagnosticando Erros de Conversão de Dados	635
Resolução de problemas de serviços da web JSON	637
Resolução de problemas de implementação do	
JSON	637
Resolução de Problemas do assistente JSON	638
Resolvendo problemas com solicitações do	
JSON	639
Respostas de erro retornadas ao cliente.	640
Códigos de retorno do assistente JSON.	641

Apêndice. Amostras de Serviços da Web 643

O Aplicativo de Exemplo do Gerenciador de	
Catálogo do CICS	643
O Aplicativo Base	644
Instalando e Configurando o Aplicativo Base	646
Executando o Aplicativo de Exemplo com a	
Interface BMS	652
Suporte de Serviço da Web para o Aplicativo de	
Exemplo	654
Configurando o Web Client.	667
Executando o Aplicativo Ativado para Serviço	
da Web	668
Implementando o Aplicativo de Exemplo	673
Componentes do Aplicativo Base.	678
Estruturas e Definições do Arquivo	686
Amostras JSON.	688
Solicitação HTTP GET de exemplo usando uma	
sequência de consultas	688
Exemplo de solicitação de HTTP com um corpo	
JSON	688

Avisos 691

Índice Remissivo 697

Sobre este PDF

Este PDF descreve como usar serviços da web no CICS. Ele é destinado a programadores de sistema que são responsáveis por configurar o CICS para suportar serviços da web e a desenvolvedores de aplicativos que são responsáveis por aplicativos que serão implementados em um ambiente de serviços da web. Antes do CICS TS V5.4, esse PDF era chamado de "Guia de Serviços da Web".

Para obter detalhes sobre os termos e notação usados, consulte Convenções e terminologia usadas na documentação do CICS no IBM Knowledge Center.

Data deste PDF

Este PDF foi criado em 18 de abril de 2017.

Capítulo 1. CICS e Serviços da Web

O CICS fornece suporte para serviços da web.

O Que é um Serviço da Web?

Um serviço da web possui uma interface, que oculta os detalhes de implementação para que ele possa ser usado independentemente da plataforma de hardware ou software na qual ele é implementado e independente da linguagem de programação na qual ele é gravado. Esta independência encoraja aplicativos baseados em serviço da web a serem implementações fracamente acopladas, orientadas a componente, de tecnologia cruzada. Os serviços da web podem ser usados sozinhos ou com outros serviços da web para realizar uma agregação complexa ou uma transação de negócios.

Serviços da web suportados pelo CICS

O CICS suporta dois protocolos de serviço da web diferentes, os protocolos SOAP e JavaScript Object Notation (JSON). Esses dois protocolos têm características e vantagens diferentes.

Terminologia de Serviços da Web

Linguagem de marcação extensível (XML)

Um padrão para a marcação do documento, que utiliza uma sintaxe genérica para os dados de marcação com tags simples legíveis. O padrão é endossado pelo World Wide Web Consortium (W3C).

Emissor SOAP Inicial

O remetente SOAP que origina uma mensagem SOAP no ponto inicial de um caminho da mensagem SOAP.

JavaScript Object Notation (JSON)

Um formato de intercâmbio de dados simples que é baseado na notação literal de objeto de JavaScript. O JSON é uma linguagem de programação neutra, mas utiliza convenções de linguagens que incluem C, C++, C#, Java™, JavaScript, Perl, Python.

esquema JSON

Um documento JavaScript Object Notation que descreve a estrutura e restringe o conteúdo de outros documentos JSON.

RESTful

Relativo aos aplicativos e serviços que estão em conformidade com as restrições do Representational State Transfer (REST).

Provedor de Serviços

A coleção de software que fornece um serviço da web.

Aplicativo do provedor de serviços

Um aplicativo que é usado em um provedor de serviços. Geralmente, um aplicativo do provedor de serviços fornece o componente de lógica de negócios de um provedor de serviços.

Solicitante de serviços

A coleção de software que é responsável por solicitar um serviço da web a partir de um provedor de serviços.

Aplicativo do solicitante de serviço

Um aplicativo que é usado em um solicitante de serviço. Geralmente, um aplicativo do solicitante de serviço fornece o componente da lógica de negócios de um solicitante de serviço.

Simple Object Access Protocol

Consulte SOAP.

SOAP Antigamente um acrônimo para *Simple Object Access Protocol*. Um protocolo leve para troca de informações em um ambiente distribuído descentralizado. Ele é um protocolo baseado em XML que consiste em três partes:

- Um envelope que define uma estrutura para descrever o que está em uma mensagem e como processá-la
- Um conjunto de regras de codificação para expressar instâncias de tipos de dados definidos pelo aplicativo
- Uma convenção para representar chamadas e respostas de procedimento remoto

SOAP pode ser utilizado com outros protocolos, como HTTP.

A especificação para SOAP 1.1 é publicada em Simple Object Access Protocol (SOAP) 1.1.

A especificação para SOAP 1.2 é publicada aqui:

SOAP Version 1.2 Part 0: Primer

SOAP Version 1.2 Part 1: Messaging Framework

SOAP Version 1.2 Part 2: Adjuncts

Intermediário do SOAP

Um nó SOAP que é um destinatário SOAP e um emissor SOAP e é destinável a partir de uma mensagem SOAP. Ele processa os blocos de cabeçalho SOAP direcionados a ele e encaminha uma mensagem SOAP para um destinatário SOAP final.

caminho da mensagem SOAP

O conjunto de nós SOAP por meio do qual uma mensagem SOAP única é transmitida. Esses nós incluem o emissor SOAP inicial, zero ou mais intermediários SOAP e um destinatário SOAP final.

Nó SOAP

Lógica de processamento que opera em uma mensagem SOAP.

Receptor SOAP

Um nó SOAP que aceita uma mensagem SOAP.

Emissor SOAP

Um nó SOAP que transmite uma mensagem SOAP.

Receptor SOAP Final

O receptor SOAP que é um destino final de uma mensagem SOAP. Ele é responsável por processar o conteúdo do corpo SOAP e de quaisquer blocos de cabeçalhos SOAP direcionados a ele.

UDDI Consulte Universal Description, Discovery and Integration.

Universal Description, Discovery and Integration

Universal Description, Discovery and Integration (UDDI) é uma especificação para registros de informações baseadas na web distribuídos de serviços da web. O UDDI também é um conjunto acessível publicamente de implementações da especificação que permitem que os

negócios registrem informações sobre os serviços da web que eles oferecem, de forma que outros negócios possam localizá-los. A especificação é publicada pelo OASIS.

Serviço da Web

Um sistema de software projetado para suportar a interação de máquina para máquina interoperável através de uma rede. Ele possui uma interface descrita em um formato processável pela máquina (especificamente, Web Service Description Language ou WSDL).

Web Services Addressing

O Web Services Addressing (WS-Addressing) fornece um mecanismo neutro de transporte para endereçar serviços da web e mensagens.

As especificações para WS-Addressing são publicadas aqui:

- Web Services Addressing 1.0 - Core
- Web Services Addressing 1.0 - SOAP Binding
- Web Services Addressing 1.0 - Metadados
- Web Services Addressing- Submission

Web Services Atomic Transaction

Uma especificação que fornece a definição de um tipo de coordenação de transação atômica utilizado para coordenar atividades com uma propriedade "tudo ou nada".

A especificação é publicada por OASIS em Web Services Atomic Transaction.

Arquivo de Ligação de Serviços da Web

Um arquivo, associado a um recurso WEBSERVICE, que contém informações que o CICS utiliza para mapear os dados entre as mensagens de entrada e saída e estruturas de dados do aplicativo.

Descrição do Serviço da Web

Um documento XML pelo qual um provedor de serviços comunica as especificações para chamar um serviço da web para um solicitante de serviços. As descrições de serviços da web são gravadas em Web Service Description Language (WSDL).

Web Service Description Language

Um aplicativo XML para descrever serviços da web. Ele foi projetado para separar as descrições das funções abstratas oferecidas por um serviço e os detalhes concretos de um serviço, por exemplo, como e onde essa função é oferecida.

A especificação é publicada em Web Services Description Language (WSDL).

Segurança de serviços da web

Um conjunto de aprimoramentos para o sistema de mensagens SOAP que fornece integridade e confidencialidade da mensagem. A especificação é publicada pelo OASIS em Web Services Security: SOAP Message Security 1.0 (WS-Security 2004).

WS-Atomic Transaction

Consulte Web Services Atomic Transaction.

WS-I Basic Profile

Um conjunto de especificações de serviços da web não proprietárias, com esclarecimentos e aperfeiçoamentos para essas especificações que, juntos, promovem a interoperabilidade entre diferentes implementações de

serviços da web. O perfil é definido pela Web Services Interoperability Organization (WS-I) e a versão 1.0 está disponível em Web Services Interoperability Organization (WS-I) Basic Profile 1.0.

WSDL

Consulte Web Service Description Language.

WSS Consulte Web Services Security.

XML Linguagem de Marcação Extensível.

As especificações para XML são publicadas aqui:

SOAP Version 1.2 Part 0: Primer

SOAP Version 1.2 Part 1: Messaging Framework

SOAP Version 1.2 Part 2: Adjuncts

Namespace XML

Uma coleção de nomes, identificados por uma referência de URI, que são usados em documentos XML como tipos de elemento e nomes do atributo.

Esquema XML

Um documento XML que descreve a estrutura e restringe o conteúdo de outros documentos XML.

Linguagem de Definição de Esquema XML

Uma sintaxe de XML para gravar esquemas XML, recomendados pelo World Wide Web Consortium (W3C).

CICS e serviços da web SOAP

O CICS suporta duas abordagens diferentes para a implementação de seus aplicativos CICS em um ambiente de serviços da web. Uma abordagem ativa a implementação rápida, com menor quantidade de esforço de programação; a outra abordagem fornece flexibilidade e controle completos sobre seus aplicativos de serviço da web, usando código que você grava para adequar às suas necessidades específicas. Ambas as abordagens são suportadas por uma infraestrutura que consiste em um ou mais programas de *pipelines* e *manipulador de mensagem* que operam em solicitações e respostas de serviços da web.

Ao implementar seus aplicativos CICS em um ambiente de serviços da web, é possível escolher dentre as opções a seguir:

- Use o assistente de serviços da web do CICS para ajudá-lo a implementar um aplicativo com a menor quantidade de esforço de programação.

Por exemplo, se desejar expor um aplicativo existente como um serviço da web, é possível iniciar com uma estrutura de dados de linguagem de alto nível e gerar a descrição de serviços da web. Alternativamente, se desejar se comunicar com um serviço da web existente, será possível iniciar com sua descrição de serviços da web e gerar uma estrutura de linguagem de alto nível que possa usar em seu programa.

O assistente de serviços da web do CICS também gera os recursos do CICS necessários para implementar seu aplicativo. E, quando seu aplicativo é executado, o CICS transforma seus dados do aplicativo em uma mensagem SOAP na saída e transforma a mensagem SOAP de volta em dados do aplicativo na entrada.

- Tenha controle completo sobre o processamento de seus dados gravando seu próprio código para mapear entre seus dados do aplicativo e a mensagem que flui entre o solicitante e o provedor de serviço.

Por exemplo, se desejar usar mensagens não SOAP dentro da infraestrutura de serviço da web, será possível gravar seu próprio código para transformar entre o formato da mensagem e o formato usado por seu aplicativo.

Qualquer que seja a abordagem seguida, é possível usar seus próprios manipuladores de mensagem para executar o processamento adicional em suas mensagens de solicitação e resposta ou usar manipuladores de mensagem fornecidos pelo CICS que são projetados especialmente para ajudá-lo a processar mensagens SOAP.

Nós SOAP

Um nó SOAP é a lógica de processamento que opera em uma mensagem SOAP.

Um nó SOAP pode executar estas operações:

- Transmitir uma mensagem SOAP
- Receber uma mensagem SOAP
- Processar uma mensagem SOAP
- Retransmitir uma mensagem SOAP

Um nó SOAP pode ser um destes tipos:

Emissor SOAP

Um nó SOAP que transmite uma mensagem SOAP.

Receptor SOAP

Um nó SOAP que aceita uma mensagem SOAP.

Emissor SOAP Inicial

O emissor SOAP que origina uma mensagem SOAP no ponto de início de um caminho de mensagem SOAP.

Intermediário do SOAP

Um intermediário do SOAP é um receptor SOAP e um emissor SOAP, destinado a partir de uma mensagem SOAP. Ele processa os blocos de cabeçalho SOAP direcionados a ele e age para encaminhar uma mensagem SOAP para um receptor SOAP final.

Receptor SOAP Final

O receptor SOAP que é um destino final de uma mensagem SOAP. Ele processa o conteúdo do corpo SOAP e quaisquer blocos de cabeçalho SOAP direcionados a ele. Em algumas circunstâncias, uma mensagem SOAP não pode atingir um receptor SOAP final; por exemplo, devido a um problema em um intermediário do SOAP.

Um Pipeline do Provedor de Serviços

Em um pipeline do provedor de serviços, o CICS recebe uma solicitação, a qual é transmitida por meio de um pipeline ao programa aplicativo de destino. A resposta do aplicativo é retornada ao solicitante de serviço por meio do mesmo pipeline.

Quando o CICS está na função de provedor de serviços, ele executa as operações a seguir:

1. Receber a solicitação do solicitante de serviço.
2. Examinar a solicitação e extrair o conteúdo que é relevante para o programa aplicativo de destino.
3. Chamar o programa de aplicativo, transmitindo dados extraídos da solicitação.
4. Quando o programa de aplicativo retornar o controle, construir uma resposta, usando dados retornados pelo programa de aplicativo.

5. Enviar uma resposta ao solicitante de serviço.

Figura 1 ilustra um pipeline de três manipuladores de mensagem em uma configuração do provedor de serviços:

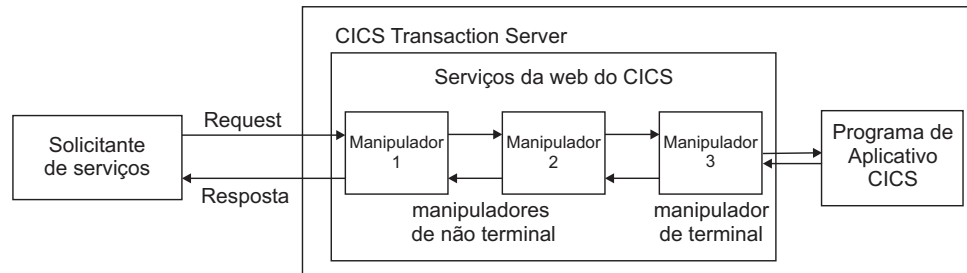


Figura 1. Um Pipeline do Provedor de Serviços

1. O CICS recebe uma solicitação do solicitante de serviço. Ele transmite a solicitação ao manipulador de mensagem 1.
2. O manipulador de mensagem 1 executa algum processamento e transmite a solicitação ao manipulador 2 (para ser preciso, ele retorna o controle ao CICS, que gerencia o pipeline. O CICS, então, passa o controle ao próximo manipulador de mensagem).
3. O manipulador de mensagem 2 recebe a solicitação do manipulador 1, executa algum processamento e transmite a solicitação ao manipulador 3.
4. O manipulador de mensagem 3 é o manipulador de terminal do pipeline. Ele usa as informações na solicitação para chamar o programa de aplicativo. Em seguida, ele usa a saída do programa de aplicativo para gerar uma resposta, a qual ele transmite de volta ao manipulador 2.
5. O manipulador de mensagem 2 recebe a resposta do manipulador 3, executa algum processamento e o transmite ao manipulador 1.
6. O manipulador de mensagem 1 recebe a resposta do manipulador 2, executa algum processamento e retorna a resposta ao solicitante de serviço.

Um Pipeline do Solicitante de Serviço

Em um pipeline do solicitante de serviço, um programa de aplicativo cria uma solicitação, que é transmitida por meio de um pipeline ao provedor de serviços. A resposta do provedor de serviços é retornada ao programa de aplicativo por meio do mesmo pipeline.

Quando o CICS está na função do solicitante de serviço, ele executa as operações a seguir:

1. Usar dados fornecidos pelo programa de aplicativo para construir uma solicitação.
2. Enviar a solicitação ao provedor de serviços.
3. Receber uma resposta do provedor de serviços.
4. Examinar a resposta e extrair o conteúdo que é relevante ao programa de aplicativo original.
5. Retornar o controle ao programa de aplicativo.

Figura 2 ilustra um pipeline de três manipuladores de mensagem em uma configuração do solicitante de serviço:

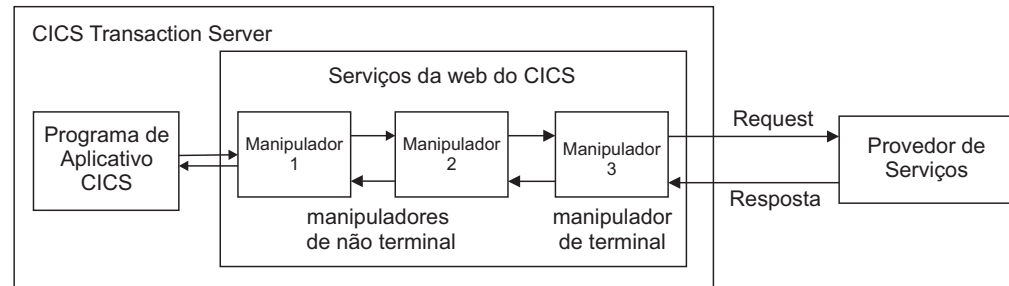


Figura 2. Um Pipeline do Solicitante de Serviço

1. Um programa de aplicativo cria uma solicitação.
2. O manipulador de mensagem 1 recebe a solicitação do programa de aplicativo, executa algum processamento e transmite a solicitação ao manipulador 2 (para ser preciso, ele retorna o controle ao CICS, que gerencia o pipeline. O CICS, então, passa o controle ao próximo manipulador de mensagem).
3. O manipulador de mensagem 2 recebe a solicitação do manipulador 1, executa algum processamento e transmite a solicitação ao manipulador 3.
4. O manipulador de mensagem 3 recebe a solicitação do manipulador 2, execute algum processamento e transmite a solicitação ao provedor de serviços.
5. O manipulador de mensagem 3 recebe a resposta do provedor de serviços, executa algum processamento e a transmite ao manipulador 2.
6. O manipulador de mensagem 2 recebe a resposta do manipulador 3, executa algum processamento e o transmite ao manipulador 1.
7. O manipulador de mensagem 1 recebe a resposta do manipulador 2, executa algum processamento e retorna a resposta ao programa de aplicativo.

Pipelines CICS e SOAP

O pipeline que o CICS usa para processar solicitações e respostas de serviço da web é genérico, pelo fato de que há algumas restrições em qual processamento pode ser executado em cada manipulador de mensagem. No entanto, muitos aplicativos de serviço da web usam mensagens SOAP e qualquer processamento dessas mensagens deve estar em conformidade com a especificação SOAP. Portanto, o CICS fornece programas de *manipulador de mensagem SOAP* especiais que podem ajudá-lo a configurar seu pipeline como um nó SOAP.

- Um pipeline pode ser configurado para uso em um solicitante de serviço ou em um provedor de serviços:
 - Um pipeline do solicitante de serviço é o emissor SOAP inicial para a solicitação e o receptor SOAP final para a resposta
 - Um pipeline do provedor de serviços é o receptor SOAP final para a solicitação e o emissor SOAP inicial para a resposta

Não é possível configurar um pipeline do CICS para funcionar como um intermediário do SOAP.

- Um pipeline do solicitante de serviço pode ser configurado para suportar SOAP 1.1 ou SOAP 1.2, mas não ambos. No entanto, um pipeline do provedor de serviços pode ser configurado para suportar SOAP 1.1 e SOAP 1.2. Dentro de seu sistema CICS, é possível ter muitos pipelines, alguns dos quais suportam SOAP 1.1 ou SOAP 1.2 e alguns dos quais suportam ambos.

- É possível configurar um pipeline do CICS para ter mais de um manipulador de mensagem SOAP.
- Os manipuladores de mensagem SOAP fornecidos pelo CICS podem ser configurados para chamar uma ou mais rotinas de manipulação do cabeçalho gravadas pelo usuário.
- Os manipuladores da mensagem SOAP fornecidos pelo CICS podem ser configurados para impingir alguns aspectos de conformidade com o WS-I Basic Profile Versão 1.1 e para impingir a presença de manipuladores específicos na mensagem SOAP.

Os manipuladores de mensagem SOAP, e suas rotinas de manipulação de cabeçalho, são especificados no arquivo de configuração de pipeline.

O Caminho da Mensagem SOAP

O caminho da mensagem SOAP é o conjunto de nós SOAP por meio dos quais uma única mensagem SOAP passa, incluindo o emissor SOAP inicial, zero ou mais intermediários SOAP e um destinatário SOAP final

No caso mais simples, uma mensagem SOAP é transmitida entre dois nós; ou seja, de um *emissor SOAP* para um *destinatário SOAP*. No entanto, em casos mais complexos, as mensagens podem ser processadas pelos nós *intermediários SOAP*, que recebem uma mensagem SOAP e, em seguida, a enviam para o próximo nó. Figura 3 mostra um exemplo de um caminho de mensagem SOAP, no qual uma mensagem SOAP é transmitida a partir do nó do emissor SOAP inicial para o nó do destinatário SOAP final, passando pelos dois nós destinatários SOAP em sua rota.

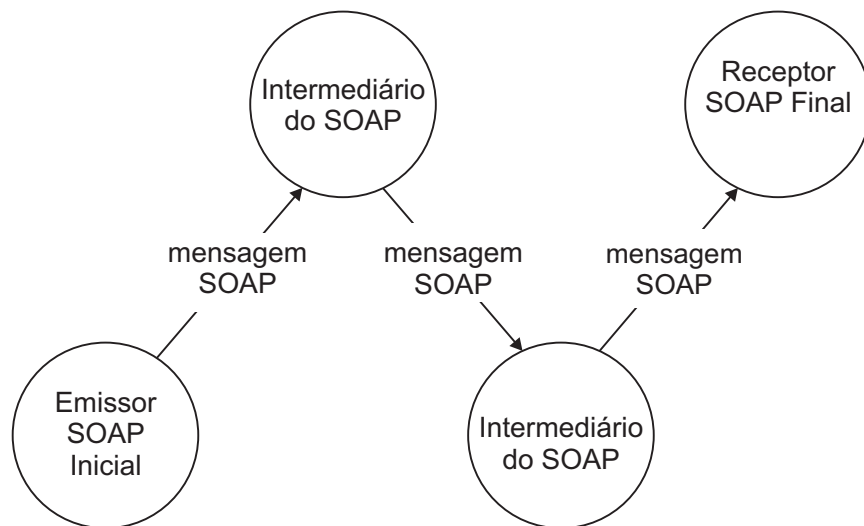


Figura 3. Um Exemplo de um Caminho de Mensagem SOAP

Um intermediário SOAP é um destinatário SOAP e um emissor SOAP. Ele pode, e em alguns casos deve, processar os blocos de cabeçalho na mensagem SOAP e encaminha a mensagem SOAP em direção ao seu destinatário final.

O *destinatário SOAP final* é o destino final de uma mensagem SOAP. Assim como o processamento dos blocos de cabeçalho, ele processa o corpo SOAP. Em algumas

circunstâncias, uma mensagem SOAP não pode atingir um receptor SOAP final; por exemplo, devido a um problema em um intermediário do SOAP.

Mensagens SOAP e a Estrutura de Dados do Aplicativo

Em muitos casos, o assistente de serviços da web do CICS pode gerar o código para transformar os dados entre uma estrutura de dados de alto nível usada em um programa aplicativo e o conteúdo do elemento <Body> de uma mensagem SOAP. Nestes casos, quando você grava seu programa de aplicativo, não é necessário analisar ou construir o corpo SOAP; o CICS fará isto para você.

Para transformar os dados, o CICS precisa de informações, no tempo de execução, sobre a estrutura de dados do aplicativo e sobre o formato das mensagens SOAP. Essas informações são mantidas em dois arquivos:

- O Arquivo de Ligação de Serviço da Web

Este arquivo é gerado pelo assistente de serviços da Web do CICS a partir de uma estrutura de dados de linguagem do aplicativo, utilizando o programa utilitário DFHLS2WS ou a partir de uma descrição de serviço da Web, utilizando o programa utilitário DFHWS2LS. O CICS usa o arquivo de ligação para gerar os recursos usados pelo aplicativo de serviço da web e para executar o mapeamento entre a estrutura de dados do aplicativo e as mensagens SOAP.

- A descrição de serviços da web

Ela pode ser uma descrição de serviço da web existente ou pode ser gerada a partir de uma estrutura de dados de linguagem do aplicativo, utilizando o programa utilitário DFHLS2WS. O CICS utiliza a descrição de serviço da web para executar a validação completa de mensagens SOAP.

Figura 4 mostra onde estes arquivos são usados em um provedor de serviços.

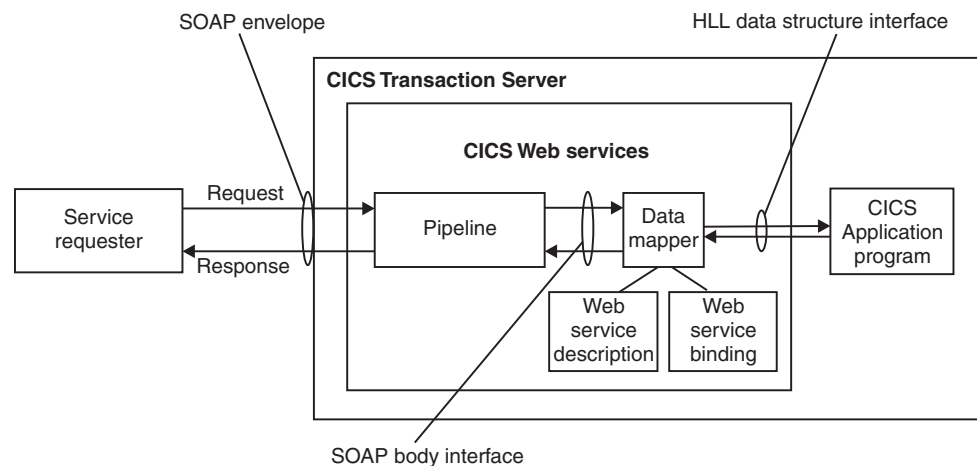


Figura 4. Mapeando o Corpo SOAP para a Estrutura de Dados do Aplicativo em um Provedor de Serviços

Um manipulador de mensagem no pipeline (geralmente, um manipulador de mensagem SOAP fornecido pelo CICS) remove o envelope SOAP de uma solicitação de entrada e transmite o corpo SOAP para a função de mapeador de dados. Ele usa o arquivo de ligação de serviço da web para mapear o conteúdo do corpo SOAP para a estrutura de dados do aplicativo. Se a validação completa da

mensagem SOAP estiver ativa, o corpo SOAP será validado em relação à descrição do serviço da web. Se houver uma resposta de saída, o processo será revertido.

Figura 5 mostra onde estes arquivos são usados em um solicitante de serviço.

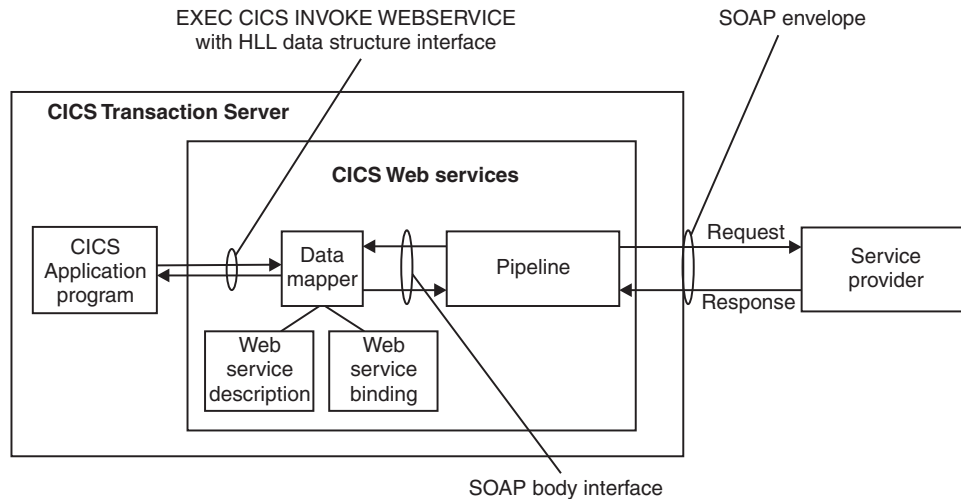


Figura 5. Mapeando o Corpo SOAP para a Estrutura de Dados do Aplicativo em um Solicitante de Serviço

Para uma solicitação de saída, a função de mapeador de dados constrói um corpo SOAP a partir da estrutura de dados do aplicativo, utilizando informações a partir do arquivo de ligação de serviço da web. Um manipulador de mensagem no pipeline (geralmente, um manipulador de mensagens SOAP fornecido pelo CICS) inclui o envelope SOAP. Se houver uma resposta de entrada, o processo será revertido. Se a validação completa da mensagem SOAP estiver ativa, o corpo SOAP de entrada será validado com relação à descrição de serviços da web.

Em ambos os casos, o ambiente de execução que permite que um programa de aplicativo CICS específico opere em uma configuração de serviços da web é definido por três objetos. Eles são o pipeline, o arquivo de ligação de serviço da web e a descrição de serviços da web. Os três objetos são definidos para o CICS como atributos da definição de recurso WEBSERVICE.

Há algumas situações nas quais, embora você esteja usando mensagens SOAP, não é possível usar a transformação que o assistente de serviços da web do CICS gera:

- Quando os mesmos dados não podem ser representados na mensagem SOAP e na linguagem de alto nível.

Todas as linguagens de alto nível que o CICS suporta, e o Esquema XML, suportam uma variedade de diferentes tipos de dados. No entanto, não há uma correspondência de um para um entre os tipos de dados utilizados nas linguagens de alto nível e aqueles utilizados no Esquema XML e há casos em que os dados podem ser representados em um, mas não no outro. Nessas situações, você deve considerar um dos procedimentos a seguir:

- Altere a estrutura de dados do aplicativo. Isso pode não ser possível, pois pode envolver alterações no próprio programa de aplicativo.
- Construir um programa wrapper, que transforma os dados do aplicativo em um formato que o CICS pode, então, transformar em um corpo da mensagem

SOAP. Se você fizer isso, poderá deixar o programa de aplicativo inalterado. Neste caso o suporte de serviço da Web do CICS interage diretamente com o programa wrapper e somente indiretamente com o programa de aplicativo.

- Quando seu programa de aplicativo estiver em uma linguagem que não é suportado pelo assistente de serviços da web do CICS.

Nesta situação, você deve considerar um dos seguintes:

- Construir um programa wrapper que é gravado em uma das linguagens que o assistente de serviços da web do CICS suporta (COBOL, PL/I, C ou C++).
- Em vez de usar o assistente de serviços da web do CICS, grave seu próprio programa para executar o mapeamento entre as mensagens SOAP e a estrutura de dados do programa de aplicativo.

WSDL e a Estrutura de Dados do Aplicativo

Uma descrição de serviços da web contém representações abstratas das mensagens de entrada e saída usadas pelo serviço. O CICS usa a descrição de serviços da web para construir as estruturas de dados usadas por programas de aplicativo. No tempo de execução, o CICS executa o mapeamento entre as estruturas de dados do aplicativo e as mensagens.

A descrição de um serviço da web contém, entre outras coisas:

- Uma ou mais operações
- Para cada operação, uma mensagem de entrada e uma mensagem de saída opcional
- Para cada mensagem, a estrutura da mensagem, definidas em termos de tipos de dados XML. Os tipos de dados complexos usados nas mensagens são definidos em um esquema XML que está contido no elemento `<types>` dentro da descrição de serviços da web. Mensagens simples podem ser descritas sem usar o elemento `<types>`.

O WSDL contém uma definição abstrata de uma operação e as mensagens associadas; ele não pode ser usado diretamente em um programa de aplicativo. Para implementar a operação, um provedor de serviços deve fazer o seguinte:

- Ele deve analisar o WSDL, para entender a estrutura das mensagens
- Ele deve analisar cada mensagem de entrada e construir a mensagem de saída
- Ele deve executar os mapeamentos entre o conteúdo de mensagens de entrada e de saída e as estruturas de dados utilizadas no programa de aplicativo

Um solicitante de serviços deve fazer o mesmo para chamar a operação.

Quando você usa o assistente de serviços da web do CICS, muito disto é feito para você e é possível gravar seu programa de aplicativo sem entendimento detalhado de WSDL ou da maneira como as mensagens de entrada e de saída são construídas.

O assistente de serviços da web do CICS consiste em dois programas utilitários:

DFHWS2LS

Este programa utilitário utiliza uma descrição de serviços da web como um ponto de início. Ele usa as descrições das mensagens, e os tipos de dados usados nessas mensagens, para construir estruturas de dados de linguagem de alto nível que você pode usar em seus programas de aplicativo.

DFHLS2WS

Este programa utilitário utiliza uma estrutura de dados de linguagem de

alto nível como um ponto de início. Ele usa a estrutura para construir uma descrição de serviços da web que contém descrições de mensagens e os tipos de dados usados nessas mensagens derivadas da estrutura de linguagem.

Ambos os programas utilitários geram um arquivo de ligação de serviços da web que o CICS usa no tempo de execução para executar o mapeamento entre as estruturas de dados do programa de aplicativo e as mensagens SOAP.

Um Exemplo de Mapeamento de COBOL para WSDL

Este exemplo mostra como a estrutura de dados usada em um programa COBOL é representada na descrição de serviços da web que é gerada pelo assistente de serviços da web do CICS.

Figura 6 mostra uma estrutura de dados COBOL simples:

```
* Catalogue COMMAREA structure
  03 CA-REQUEST-ID          PIC X(6).
  03 CA-RETURN-CODE         PIC 9(2).
  03 CA-RESPONSE-MESSAGE    PIC X(79).
* Campos Utilizados em 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. Definição de Registro COBOL de uma Mensagem de Entrada Definida no WSDL

Os elementos chave no fragmento correspondente da descrição de serviços da web são mostrados em Figura 7 na 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 WSDL Derivado de uma Estrutura de Dados COBOL

WSDL e Message Exchange Patterns

Um documento WSDL 2.0 contém um padrão de troca de mensagem (MEP) que define a maneira como mensagens SOAP 1.2 são trocadas entre o solicitante de serviço da web e o provedor de serviço da web.

O CICS suporta quatro de oito padrões de troca de mensagens que são definidos na especificação *WSDL 2.0 Part 2: Adjuncts* e na especificação *WSDL 2.0 Part 2: Additional MEPs* para aplicativos do provedor de serviços e do solicitante de serviço. As MEPs a seguir são suportadas:

In-Only

Uma mensagem de solicitação é enviada ao provedor de serviços da web, mas o provedor não tem permissão para enviar nenhum tipo de resposta ao solicitante de serviços da web.

- No modo de provedor, quando o CICS recebe uma mensagem de solicitação a partir de um serviço da web que utiliza a MEP In-Only, ele não retorna uma mensagem de resposta. O contêiner

DFHNORESPONSE é colocado no canal do manipulador SOAP para indicar que o pipeline não deve enviar uma mensagem de resposta.

- No modo do solicitante, o CICS envia a mensagem de solicitação para o provedor de serviços da web e não aguarda por uma resposta.

In-Out

Uma mensagem de solicitação é enviada ao provedor de serviços da web e uma mensagem de resposta é retornada ao solicitante de serviços da web. A mensagem de resposta pode ser uma mensagem de SOAP normal ou uma falha do SOAP.

- No modo de provedor, quando o CICS recebe uma mensagem de solicitação de um serviço da web que usa a MEP In-Out, ele retorna uma mensagem de resposta ao solicitante.
- No modo do solicitante, o CICS envia uma mensagem de solicitação e aguarda uma resposta. Essa resposta é uma mensagem de resposta normal ou uma mensagem de falha de SOAP. A duração de tempo que o CICS espera por uma resposta é configurada no pipeline e se aplica a todos os serviços da web que usam esse pipeline. Se a solicitação atingir o tempo limite antes do CICS receber uma resposta, um erro será retornado ao aplicativo do solicitante de serviços.

In-Optional-Out

Uma mensagem de solicitação é enviada ao provedor do serviço da web e uma mensagem de resposta é, opcionalmente, retornada ao solicitante de serviço da web. A mensagem de resposta pode ser tanto uma mensagem de SOAP normal como uma falha do SOAP.

- No modo de provedor, a decisão sobre se deve retornar uma mensagem de resposta SOAP, uma falha de SOAP ou nenhuma resposta, ocorre no tempo de execução e é dependente da lógica de aplicativo do provedor de serviços. Se o CICS não enviar uma resposta ao solicitante de serviço da web, o contêiner DFHNORESPONSE será colocado no canal do manipulador SOAP para indicar que o pipeline não deve enviar uma mensagem de resposta. Se nenhuma mensagem for enviada, o aplicativo do provedor de serviços deverá excluir o contêiner DFHWS-DATA do canal.
- No modo do solicitante, o CICS envia uma mensagem de solicitação e aguarda por uma resposta do solicitante de serviço da web. Se a solicitação atingir o tempo limite antes de uma resposta ser recebida, o CICS assume que a mensagem foi recebida com sucesso e que o provedor não precisou enviar uma resposta. A duração de tempo que o CICS espera por uma resposta é configurada no pipeline e se aplica a todos os serviços da web que usam esse pipeline.

In-Only Robusto

Uma mensagem de solicitação é enviada ao provedor de serviço da web e uma mensagem de resposta é retornada somente ao solicitante de serviço da web se um erro ocorrer. Se houver um erro, uma mensagem de falha do SOAP é enviada ao solicitante.

- No modo de provedor, se o pipeline transmitir a mensagem de solicitação com sucesso ao aplicativo, um contêiner DFHNORESPONSE será colocado no canal do manipulador SOAP para indicar que o pipeline não deve enviar uma mensagem de resposta. Se ocorrer um erro no pipeline, uma mensagem de falha de SOAP será retornada ao solicitante.
- No modo do solicitante, o CICS envia a mensagem de solicitação ao provedor de serviço da web e aguarda por um período especificado

antes de atingir o tempo limite. A duração de tempo que o CICS espera por uma resposta é configurada no pipeline e se aplica a todos os serviços da web que usam esse pipeline. Se houver um tempo limite, o CICS assume que a mensagem de solicitação foi recebida com sucesso.

Para obter mais informações sobre padrões de troca de mensagem no WSDL 2.0, consulte as especificações de W3C a seguir:

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

O Arquivo de Ligação de Serviço da Web

O *arquivo de ligação de serviço da web* contém informações que o CICS usa para mapear dados entre mensagens de entrada e saída e estruturas de dados do aplicativo.

Uma descrição de serviços da web contém representações abstratas das mensagens de entrada e saída usadas pelo serviço. Quando um aplicativo provedor de serviços ou solicitante de serviço é executado, o CICS precisa de informações sobre como o conteúdo das mensagens é mapeado para as estruturas de dados usadas pelo aplicativo. Estas informações são mantidas em um arquivo de ligação de serviço da web.

Arquivos de ligação de serviço da web são criados:

- Pelo programa utilitário DFHWS2LS quando estruturas de linguagem são geradas a partir do WSDL.
- Pelo programa utilitário DFHLS2WS quando o WSDL é gerado a partir de uma estrutura de linguagem.

No tempo de execução, o CICS usa informações no arquivo de ligação de serviços da web para executar o mapeamento entre estruturas de dados do aplicativo e mensagens SOAP. Arquivos de ligação de serviço da Web são definidos no CICS no atributo WSBIND do recurso WEBSERVICE.

Normas Externas

O suporte do CICS para serviços da web está em conformidade com inúmeros padrões de mercado e especificações.

Arquitetura SOAP e formato da mensagem

SOAP é um protocolo para a troca de informações em um ambiente distribuído. As mensagens SOAP são codificadas como documentos XML e podem ser trocadas usando vários protocolos subjacentes.

Antigamente um acrônimo para *Simple Object Access Protocol*, SOAP foi desenvolvido pelo World Wide Web Consortium (W3C) e foi definido nos documentos a seguir emitidos pelo W3C. Consulte estes documentos para obter informações completas, e oficiais, sobre SOAP.

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

SOAP Version 1.2 Part 0: Primer (Recomendação do W3C)

SOAP Version 1.2 Part 1: Messaging Framework (Recomendação do W3C)

SOAP Version 1.2 Part 2: Adjuncts (Recomendação do W3C)

A especificação SOAP descreve um modelo de processamento distribuído no qual uma *mensagem SOAP* é transmitida entre *nós SOAP*. A mensagem se origina em um *emissor SOAP* e é enviada para um *destinatário SOAP*. Entre o emissor e o destinatário, a mensagem pode ser processada por um ou mais *intermediários SOAP*.

Uma mensagem SOAP é uma transmissão unidirecional entre nós SOAP, de um emissor SOAP para um destinatário SOAP, mas mensagens podem ser combinadas para construir interações mais complexas, tais como solicitação e resposta, e conversas ponto a ponto.

A especificação também inclui estas informações:

- Um conjunto de regras de codificação para expressar instâncias de tipos de dados definidos pelo aplicativo.
- Uma convenção para representar chamadas e respostas de procedimento remoto.

Arquitetura de serviços da web SOAP

A arquitetura de serviços da web SOAP é baseada em interações entre três componentes: um provedor de serviços, um solicitante de serviço e um registro de serviço opcional.

O provedor de serviços

A coleção de software que fornece um serviço da web.

- O programa de aplicativo
- O middleware
- A plataforma na qual eles são executados

O solicitante de serviço

A coleção de software que é responsável por solicitar um serviço da web a partir de um provedor de serviços.

- O programa de aplicativo
- O middleware
- A plataforma na qual eles são executados

O registro de serviço

O registro de serviço é um local central no qual provedores de serviços podem publicar suas descrições do serviço e onde solicitantes de serviço podem localizar essas descrições do serviço.

O registro é um componente opcional da arquitetura de serviços da web porque solicitantes e provedores de serviço podem se comunicar sem ele em muitas situações. Por exemplo, a organização que fornece um serviço pode distribuir a descrição do serviço diretamente aos números do serviço de inúmeras maneiras, incluindo oferecer o serviço como um download a partir de um site FTP.

O uso de um registro de serviço oferece inúmeras vantagens ao solicitante e ao provedor; por exemplo, o uso do IBM® WebSphere Service Registry and Repository (WSRR) pode ajudar o solicitante a localizar serviços mais rapidamente e pode ajudar o provedor a impingir o controle de versão dos serviços que estão sendo oferecidos.

O CICS fornece suporte direto para implementar componentes do solicitante de serviço e do provedor de serviços. No entanto, é necessário software adicional para implementar um registro de serviço no CICS. Se usar o IBM WebSphere Service

Registry and Repository (WSRR), o CICS fornece suporte para WSRR por meio do assistente de serviço da web. Como alternativa, é possível implementar um registro de serviço em uma outra plataforma.

Interações Entre um Provedor de Serviços, um Solicitante de Serviço e um Registro de Serviço

As interações entre o provedor de serviços, o solicitante de serviço e o registro de serviço envolvem as operações a seguir:

Publicar

Quando um registro de serviço é usado, um provedor de serviços publica sua descrição do serviço em um registro de serviço para o solicitante de serviço localizar.

find Quando um registro de serviço é usado, um solicitante de serviço localizar a descrição do serviço no registro.

Bind O solicitante de serviço usa a descrição do serviço para ligação com o provedor de serviços e interação com a implementação do serviço da web.

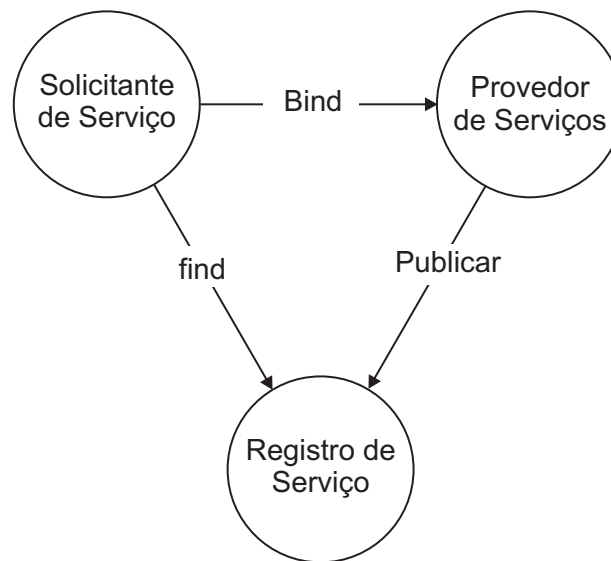


Figura 8. Componentes e Interações de Serviços da Web

Descrição do Serviço da Web:

Uma descrição de serviço da web é um documento pelo qual o *provedor de serviços* comunica as especificações para iniciar o serviço da web para o *solicitante de serviço*. As descrições de serviços da web são expressas no aplicativo XML conhecido como Web Service Description Language (WSDL).

A descrição de serviço descreve o serviço da web de tal forma a minimizar a quantidade de conhecimento compartilhado e de programação customizada que são necessários para assegurar a comunicação entre o provedor de serviços e o solicitante de serviços. Por exemplo, nem o solicitante nem o provedor precisam estar cientes da plataforma na qual o outro é executado, nem da linguagem de programação na qual o outro é gravado.

Uma descrição de serviço pode estar em conformidade com a especificação WSDL 1.1 ou WSDL 2.0. Cada um possui diferenças na terminologia e nos elementos principais que podem ser incluídos na descrição do serviço. As informações a seguir utilizam a terminologia de WSDL 1.1 e elementos para explicar o propósito da descrição do serviço.

A estrutura de WSDL permite que uma descrição de serviço seja particionada em duas definições:

- Uma *definição de Interface de Serviço* abstrata que descreve as interfaces do serviço e torna possível gravar programas que implementam e iniciam o serviço.
- Uma *definição de implementação de serviço* concreta que descreve o local na rede (ou *terminal*) do serviço da web do provedor e outros detalhes específicos da implementação. Ele permite que um solicitante de serviço se conecte ao provedor de serviços.

Consulte Figura 9 na página 19.

Um documento WSDL 1.1 utiliza os seguintes elementos principais na definição de serviços de rede:

<types>

Um contêiner para definições de tipos de dados utilizando algum sistema de tipo (como Esquema XML). Define os tipos de dados utilizados na mensagem. O elemento <types> não é necessário quando todas as mensagens consistem em tipos de dados simples.

<message>

Especifica quais tipos de dados XML são utilizados para definir os parâmetros de entrada e saída de uma operação.

<portType>

Define o conjunto de operações suportadas por um ou mais terminais. Em um elemento <portType>, cada operação é descrita por um elemento <operation>.

<operation>

Especifica quais mensagens XML podem aparecer nos fluxos de dados de entrada e saída. Uma operação é comparável a uma assinatura de método em uma linguagem de programação.

<binding>

Descreve o protocolo, o formato de dados, a segurança e outros atributos para um elemento <portType> específico.

<port>

Especifica o endereço de rede de um terminal e o associa a um elemento <binding>.

<service>

Define o serviço da web como uma coleção de terminais relacionados. Um elemento <service> contém um ou mais elementos <port>.

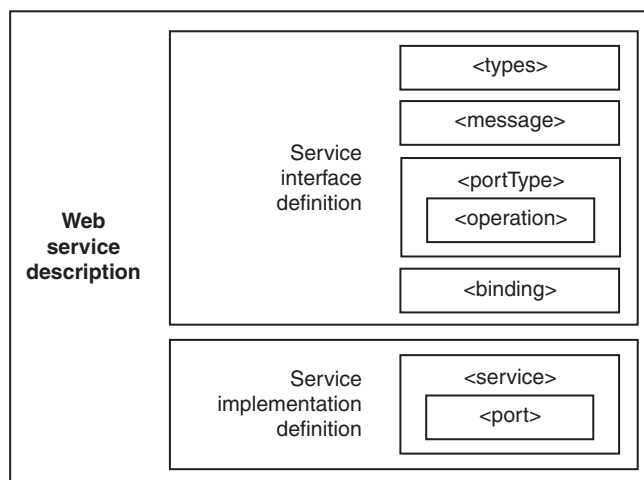


Figura 9. Estrutura de uma Descrição de Serviços da Web

Como é possível particionar a descrição de serviços da web, você pode dividir a responsabilidade pela criação de uma descrição de serviço completa. Como uma ilustração, considere um serviço que é definido por um corpo de padrões a ser utilizado em um segmento de mercado e é implementado por empresas individuais nesse segmento de mercado:

- O corpo de padrões fornece uma definição de Interface de Serviço, contendo os elementos a seguir:
 - <types>
 - <message>
 - <portType>
 - <binding>
- Um provedor de serviços que deseja oferecer uma implementação do serviço fornece uma definição de implementação de serviço, contendo os seguintes elementos:
 - <port>
 - <service>

Publicação do Serviço:

É possível publicar uma descrição do serviço usando vários mecanismos diferentes. Cada mecanismo é adequado para uso em situações diferentes. O CICS suporta o uso do IBM WebSphere Service Registry and Repository (WSRR) para publicar descrições do serviço. Como alternativa, é possível usar outros métodos para publicar uma descrição do serviço.

WSSR O CICS suporta o uso do WSRR para publicar descrições do serviço. Para obter mais informações sobre o suporte que o CICS fornece para WSSR, consulte o tópico "Interoperabilidade entre o Assistente de Serviços da Web e o WSRR" no Centro de Informações.

Qualquer um dos mecanismos a seguir, nenhum dos quais é suportado diretamente pelo CICS, pode ser usado com o CICS para publicar descrições do serviço:

Publicação Direta

Este mecanismo é o mais direto para publicar descrições do serviço; o provedor de serviços envia a descrição do serviço diretamente ao solicitante de serviço, usando um anexo de e-mail, um site FTP ou uma distribuição de CD-ROM.

DISCO

Estes protocolos proprietários fornecem um mecanismo de publicação dinâmica. O solicitante de serviço usa um mecanismo HTTP GET simples para recuperar uma descrição de serviços da web de um local de rede que é especificado pelo provedor de serviços e identificado com uma URL.

UDDI (Universal Description, Discovery and Integration)

Uma especificação para registros de informações baseados na web distribuídos de serviços da web. O UDDI também é um conjunto publicamente acessível de implementações da especificação que permitem que as empresas registrem informações sobre os serviços da web que eles oferecem para que outras empresas possam localizá-las.

Uma descrição de serviço pode ser publicada em mais de um formulário, se necessário.

Estrutura de uma Mensagem SOAP

Uma mensagem SOAP é codificada como um documento XML, consistindo em um elemento <Envelope>, que contém um elemento <Header> opcional e um elemento <Body> obrigatório. O elemento <Fault>, contido no <Body>, é utilizado para relatar erros.

O Envelope SOAP

O <Envelope> SOAP é o elemento-raiz em cada mensagem SOAP. Ele contém dois elementos filhos, um <Header> opcional e um <Body> obrigatório.

O Cabeçalho SOAP

O <Header> SOAP é um subelemento opcional do envelope SOAP. Ele é usado para transmitir informações relacionadas ao aplicativo que devem ser processadas pelos nós SOAP ao longo do caminho da mensagem.

O Corpo SOAP

O <Body> SOAP é um subelemento obrigatório do envelope SOAP. Ele contém informações destinadas ao destinatário final da mensagem.

A Falha de SOAP

<Fault> SOAP é um subelemento do corpo SOAP, que é usado para relatar erros.

Com a exceção do elemento <Fault>, que está contido no <Body> de uma mensagem SOAP, os elementos XML no <Header> e o <Body> são definidos pelos aplicativos que os utilizam. No entanto, a especificação SOAP impõe algumas restrições em suas estruturas.

Figura 10 na página 21 mostra os principais elementos de uma mensagem SOAP.

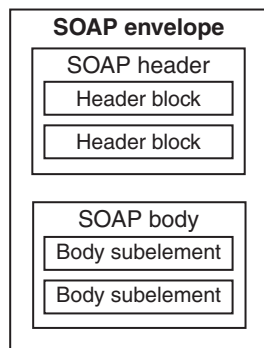


Figura 10. A Estrutura de uma Mensagem SOAP

Figura 11 é um exemplo de uma mensagem SOAP que contém blocos de cabeçalho (os elementos `<m:reservation>` e `<n:passenger>`) e um corpo (contendo os elementos `<p:itinerary>` e `<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. Um Exemplo de uma Mensagem SOAP 1.2

O Cabeçalho SOAP:

O <Header> SOAP é um elemento opcional em uma mensagem SOAP. Ele é usado para transmitir informações relacionadas ao aplicativo que devem ser processadas pelos nós SOAP ao longo do caminho da mensagem.

Os elementos filhos imediatos do elemento <Header> são chamados de blocos de cabeçalhos. Um bloco de cabeçalho é um elemento XML definido pelo aplicativo. Ele representa um agrupamento lógico de dados que podem ser destinados nos nós SOAP que podem ser encontrados no caminho de uma mensagem de um emissor para um receptor final.

Os blocos de cabeçalhos SOAP podem ser processados por nós intermediários SOAP e pelo nó do receptor SOAP final. No entanto, em um aplicativo real, nem todos os nós processam cada bloco de cabeçalho. Em vez disso, cada nó geralmente é projetado para processar blocos de cabeçalho específicos e, de modo inverso, cada bloco de cabeçalho é destinado a ser processado por nós específicos.

O cabeçalho SOAP permite que os recursos sejam incluídos em uma mensagem SOAP de uma maneira descentralizada sem antes acordar entre as partes da comunicação. SOAP define alguns atributos que podem ser usados para indicar o que lidará com um recurso e se ele é opcional ou obrigatório. Tais informações de "controle" incluem, por exemplo, transmitir diretivas ou informação contextual relacionada ao processamento da mensagem. Desta maneira, uma mensagem SOAP pode ser estendida de uma maneira específica do aplicativo.

Embora os blocos de cabeçalhos sejam definidos pelo aplicativo, os atributos definidos por SOAP nos blocos de cabeçalhos indicam como os blocos de cabeçalhos devem ser processados pelos nós SOAP. Observe estes atributos importantes:

encodingStyle

Indica as regras utilizadas para codificar as partes de uma mensagem SOAP. SOAP define um conjunto mais limitado de regras para codificação de dados do que a codificação mais flexível que o XML permite.

role (SOAP 1.2)

actor (SOAP 1.1)

No SOAP 1.2, o atributo **role** especifica se um nó específico opera em uma mensagem. Se a função especificada para o nó corresponder ao atributo role do bloco de cabeçalhos, o nó processará o cabeçalho. Se as funções não corresponderem, o nó não processará o bloco de cabeçalhos. No SOAP 1.1, o atributo **actor** possui a mesma função.

As funções podem ser definidas pelo aplicativo e são designadas por um URI. Por exemplo, `http://example.com/Log` pode designar a função de um nó que executa a criação de log. Os blocos de cabeçalho que devem ser processados por este nó especificam `env:role="http://example.com/Log"`, em que o prefixo de namespace `env` está associado ao nome do namespace SOAP de `http://www.w3.org/2003/05/soap-envelope`.

A especificação SOAP 1.2 define três funções padrão, além daquelas que são definidas pelo aplicativo:

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

Nenhum dos nós SOAP no caminho da mensagem processarão o bloco de

cabeçalho diretamente. Os blocos de cabeçalhos com esta função podem ser usados para transportar dados necessários para processamento de outros blocos de cabeçalhos SOAP.

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

Todos os nós SOAP no caminho da mensagem são esperados para examinar o bloco de cabeçalho, desde que o cabeçalho não tenha sido removido por um nó anteriormente no caminho da mensagem.

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

Somente o nó receptor final é esperado para examinar o bloco de cabeçalho.

mustUnderstand

Este atributo é usado para assegurar que os nós SOAP não ignorem blocos de cabeçalho que são importantes para o propósito geral do aplicativo. Se um nó SOAP determinar, usando o atributo **role** ou **actor**, que processará um bloco de cabeçalho, e o atributo **mustUnderstand** tiver um valor "true", o nó deverá processar o bloco de cabeçalho de uma maneira consistente com sua especificação ou não processar de maneira nenhuma (e lançar uma falha). Mas, se o atributo tiver um valor "false", o nó não será obrigado a processar o bloco de cabeçalho.

De fato, o atributo **mustUnderstand** indica se o processamento do bloco de cabeçalhos é obrigatório ou opcional.

O atributo **mustUnderstand** possui estes valores:

true (SOAP 1.2)

1 (SOAP 1.1)

O nó deve processar o bloco de cabeçalho de uma maneira consistente com sua especificação ou não processar de maneira nenhuma (e lançar uma falha).

false (SOAP 1.2)

0 (SOAP 1.1)

O nó não é obrigado a processar o bloco de cabeçalho.

relay (Apenas SOAP 1.2)

Quando um nó intermediário SOAP processa um bloco de cabeçalho, ele o remove da mensagem SOAP. Por padrão, ele também remove qualquer bloco de cabeçalho que é ignorado, porque o atributo **mustUnderstand** tinha um valor "false". No entanto, quando o atributo **relay** é especificado com um valor "true", o nó retém o bloco de cabeçalho não processado na mensagem.

O Corpo SOAP:

<Body> é o elemento obrigatório no envelope SOAP, no qual as informações de ponta a ponta principais transmitidas em uma mensagem SOAP são transportadas.

O elemento <Body> e seus elementos filhos associados são usados para trocar informações entre o emissor SOAP inicial e o destinatário SOAP final. SOAP define um elemento filho para <Body>: o elemento <Fault>, o qual é usado para relatar erros. Outros elementos no <Body> são definidos pelo serviço da web que os usa.

A Falha de SOAP:

O elemento SOAP <Fault> transporta informações de erro e status na mensagem SOAP.

Se um erro ocorrer em um serviço da web, uma mensagem de falha será retornada ao cliente. A estrutura básica da mensagem de falha é definida nas especificações SOAP. Cada mensagem de falha pode incluir XML que descreve a condição de erro específica. Por exemplo, se um encerramento de forma anormal do aplicativo ocorrer em um serviço da web do CICS, uma mensagem de falha é retornada ao cliente que relata o encerramento de forma anormal.

O CICS pode enviar diferentes tipos de mensagem de falha:

- As mensagens de falha de SOAP padrão são definidas pelas especificações de SOAP ou uma das especificações de serviço da web que são suportadas no CICS. As condições de erro comuns do relatório de falhas, tais como envelopes SOAP malformados.
- As mensagens de falha de SOAP do aplicativo são geradas usando os comandos da API **EXEC CICS SOAPFAULT** em resposta às condições que são detectadas ou manipuladas pelo aplicativo. A estrutura destas mensagens de falha é conhecida para o aplicativo, mas não para o CICS.
- As mensagens de falha do manipulador SOAP são geradas pelos programas do manipulador SOAP em resposta para manipulação de erros geral no CICS. Por exemplo, os programas do manipulador SOAP enviam falhas de SOAP para encerramentos anormais, falhas de análise de XML e outros erros comuns.
- Mensagens de falha do manipulador de aplicativo são geradas pelos manipuladores de aplicativo SOAP do CICS em resposta à descoberta de erros ao processar o corpo de uma mensagem SOAP. Estas falhas ocorrem durante o processo de transformação do XML em dados do aplicativo binários ou ao gerar a resposta.

Se ocorrer um erro, o elemento SOAP <Fault> deverá ser uma entrada do corpo e não deverá estar presente mais de uma vez em um elemento <Body>. Os elementos XML dentro do elemento SOAP <Fault> são diferentes no SOAP 1.1 e SOAP 1.2.

SOAP 1.1

No SOAP 1.1, o elemento SOAP <Fault> contém os elementos a seguir:

<faultcode>

O elemento <faultcode> é um elemento obrigatório no elemento <Fault>. Ele fornece informações sobre a falha em um formato que pode ser processado pelo software. SOAP define um pequeno conjunto de códigos de falhas do SOAP que cobrem as falhas básicas do SOAP e este conjunto pode ser estendido por aplicativos.

<faultstring>

O elemento <faultstring> é um elemento obrigatório no elemento <Fault>. Ele fornece informações sobre a falha em um formato destinado a um leitor humano.

<faultactor>

O elemento <faultactor> contém o URI do nó SOAP que gerou a falha. Um nó SOAP que não é o destinatário SOAP final deve incluir o elemento <faultactor> quando ele cria uma falha. Um receptor SOAP final não é obrigado a incluir este elemento, mas pode fazê-lo.

<detail>

O elemento <detail> transporta informações de erro específicas do aplicativo relacionadas ao elemento <Body>. Ele deve estar presente se o conteúdo do elemento <Body> não tiver sido processado com êxito. Ele não deve ser utilizado para transportar informações sobre informações de erro

pertencentes a entradas do cabeçalho. Informações de erro detalhadas pertencentes às entradas do cabeçalho devem ser transportadas em entradas do cabeçalho.

SOAP 1.2

No SOAP 1.2, o elemento SOAP <Fault> contém os elementos a seguir:

<Código>

O elemento <Code> é um elemento obrigatório no elemento <Fault>. Ele fornece informações sobre a falha em um formato que pode ser processado pelo software. Ele contém um elemento <Value> e um elemento opcional <Subcode>.

<Reason>

O elemento <Reason> é um elemento obrigatório no elemento <Fault>. O elemento <Reason> contém um ou mais elementos <Text>, cada um dos quais contém informações sobre a falha em um idioma nativo diferente.

<Node>

O elemento <Node> contém o URI do nó SOAP que gerou a falha. Um nó SOAP que não é o destinatário SOAP final deve incluir o elemento <Node> quando ele cria uma falha. Um receptor SOAP final não é obrigado a incluir este elemento, mas pode fazê-lo.

<Role>

O elemento <Role> contém um URI que identifica a função na qual o nó estava operando no ponto em que a falha ocorreu.

<Detail>

O elemento <Detail> é um elemento opcional, que contém informações de erro específicas do aplicativo relacionadas aos códigos de falha SOAP que descrevem a falha. A presença do elemento <Detail> não possui significado com relação a quais partes da mensagem SOAP com falha foram processadas.

Exemplo e Esquemas de Falha SOAP

O exemplo a seguir mostra uma mensagem de falha SOAP que é gerada pelo manipulador de aplicativos, DFHPITP, ao processar o corpo de uma mensagem SOAP.

```
<SOAP-ENV:Fault xmlns="">
  <faultcode>SOAP-ENV:Server</faultcode>
  <faultstring>Conversion to SOAP failed</faultstring>
  <detail>
    <CICSFault xmlns="http://www.ibm.com/software/http/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>
```

A maioria do conteúdo neste exemplo é comum a todas as mensagens de falha. O elemento <Detail> contém as informações exclusivas que descrevem o problema que foi encontrado pelo manipulador de aplicativos. Esta mensagem de falha específica contém uma cópia de uma mensagem de erro que é gravada nos logs de mensagens do CICS. Se você desejar analisar as falhas de SOAP do manipulador de aplicativo programaticamente, utilize o esquema XML a seguir:

```

<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.ibm.com/software/http/cics/WSFault"
xmlns:tns="http://www.ibm.com/software/http/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>

```

As mensagens de falha de propósito geral são mais complicadas porque a seção **<Detail>** pode ser estruturada de várias maneiras diferentes. Se desejar analisar falhas do manipulador SOAP programaticamente, use o esquema XML que é fornecido no *usshome/schemas/soapfault/soapfault.xsd*, em que *usshome* é o valor do parâmetro de inicialização do sistema **USSHOME**.

Planejando para usar serviços da web SOAP

Antes de poder planejar usar serviços da web SOAP no CICS, é necessário considerar estas questões para cada aplicativo.

Antes de Iniciar

Você planeja implementar seu aplicativo CICS na função de um provedor de serviços ou de um solicitante de serviço?

Você pode ter um par de aplicativos que deseja conectar usando suporte do CICS para serviços da web. Neste caso, um aplicativo será o provedor de serviços; o outro será o solicitante de serviço.

Você planeja usar seus programas de aplicativo existentes ou gravar novos?

Se seus aplicativos existentes foram projetados com uma interface bem definida para a lógica de negócios, você provavelmente poderá usá-los em uma configuração de serviços da web, como um provedor de serviços ou um solicitante de serviço. No entanto, na maioria dos casos, você precisará gravar um programa wrapper que conecta sua lógica de negócios à lógica de serviços da web.

Se planeja gravar novos aplicativos, você deverá tentar manter sua lógica de negócios separada de sua lógica de serviços da web e, mais uma vez, você precisará gravar um programa wrapper para fornecer esta separação. No entanto, se seu aplicativo for projetado com serviços da web em mente, o wrapper poderá ser mais simples de gravar.

Você pretende usar mensagens SOAP?

SOAP é fundamental para a arquitetura de serviços da web e muito do suporte que é fornecido no CICS assume que você usará SOAP. No entanto, pode haver situações nas quais você deseja usar outros formatos da mensagem. Por exemplo, você pode ter desenvolvido seus próprios formatos da mensagem que deseja implementar com a infraestrutura de serviços da web do CICS. É possível fazer isto com o CICS, mas você não poderá usar algumas das funções que o CICS fornece, tal como o assistente de serviços da web e os manipuladores de mensagem SOAP.

Se decidir não usar SOAP, seus programas de aplicativo serão responsáveis por analisar mensagens de entrada e construir mensagens de saída.

Você pretende usar o assistente de serviços da web do CICS para gerar os mapeamentos entre suas estruturas de dados e mensagens SOAP?

O assistente fornece uma implementação rápida de muitos aplicativos em uma configuração de serviços da web com pouca ou nenhuma programação adicional. E quando a programação adicional é necessária, ela geralmente é direta e pode ser feita sem alterar a lógica de negócios existente.

No entanto, há casos que são melhor manipulados sem usar o assistente de serviços da web. Por exemplo, se você tiver código existente que mapeia as estruturas de dados para mensagens SOAP, não há vantagem na reengenharia do seu aplicativo com o assistente de serviços da Web.

Embora o assistente de serviços da web do CICS suporte a maioria dos tipos de dados e estruturas comuns, há alguns que não são suportados. Nesta situação, você deve verificar a lista de tipos de dados e estruturas não suportados para o idioma em questão e considerar o fornecimento de uma camada de programa que mapeia seus dados do aplicativo para um formato que o assistente possa suportar. Se isso não for possível, você mesmo precisará analisar a mensagem. Para obter detalhes sobre o que o assistente pode e não pode suportar, consulte Linguagem de Alto Nível e Mapeamento de Esquema XML.

Se decidir não utilizar o assistente de serviços da Web do CICS, você poderá utilizar uma ferramenta tal como IBM Developer for z Systems para criar os artefatos necessários e poderá, então, fornecer seu próprio código para analisar mensagens de entrada e construir mensagens de saída. Também é possível usar a API da interface do fornecedor fornecida.

Você pretende usar uma descrição do serviço existente ou criar uma nova?

Em algumas situações, você será obrigado a usar uma descrição do serviço existente como um ponto de início. Por exemplo:

- Seu aplicativo é um solicitante de serviço e ele é projetado para chamar um serviço da web existente.
- Seu aplicativo é um provedor de serviços e você deseja que ele esteja em conformidade com uma descrição do serviço padrão de mercado existente.

Em outras situações, pode ser necessário criar uma nova descrição do serviço para seu aplicativo.

O que Fazer Depois

Serviços da web do CICS e JSON

Há várias maneiras de iniciar com os serviços da web JSON no CICS. A maneira mais apropriada para você depende de quanto você já conhece e do nível de avanço de seus planos para usar os serviços da web.

Sobre Esta Tarefa

Uma das opções para serviços da web JSON envolve o uso do z/OS Connect. Consulte Introdução ao z/OS Connect para obter mais informações sobre essa opção. Aqui estão alguns pontos de início para os serviços da web JSON no CICS:

Procedimento

- Instale o aplicativo de exemplo. O CICS fornece um exemplo de um aplicativo de gerenciamento de catálogo, que pode ser ativado como um provedor de

serviço da web JSON. Para fazer isso, use o DFHLS2JS para gerar um serviço da web das estruturas de idiomas fornecidas. É possível usar um navegador da web ou aplicativo cliente de terceiros para testar o serviço da web JSON. Para obter mais informações, consulte Criando um Aplicativo do Provedor de Serviços a Partir de uma Estrutura de Dados.

Use o aplicativo de exemplo se desejar uma maneira prática de aprender sobre serviços da web no CICS. O aplicativo de exemplo é descrito em O aplicativo de exemplo do gerenciador de catálogo do CICS.

- Planeje a implementação de um aplicativo como um provedor de serviços. Talvez você já saiba o suficiente sobre como usará os serviços da web no CICS para iniciar o planejamento de seus aplicativos e a infraestrutura relacionada.
- Se você deseja ver um exemplo de como é possível usar os serviços da web JSON, assista ao vídeo Ganhe mobilidade em suas transações no CICS Showcase. Outra opção para serviços da web JSON envolve o uso do z/OS Connect. Consulte Introdução ao z/OS Connect para obter mais informações sobre essa opção.

Conceitos de serviços da web JSON

Leia este tópico para entender os conceitos por trás dos serviços da web JSON.

Serviços da Web

Um “serviço da Web” é um termo genérico para uma função de software que é hospedada em um local endereçável da rede. Nesse sentido geral, isso pode significar um serviço baseado em Nuvem, um serviço Utilitário ou até mesmo um aplicativo departamental. O termo “serviço da web” também pode ser usado em um sentido mais específico, tal como um serviço hospedado usando SOAP que é descrito usando um documento WSDL. É esse significado mais específico que geralmente é indicado pelo termo “serviços da web no CICS”. No entanto, o termo mais geral é frequentemente usado pela comunidade JSON ao descrever serviços baseados em JSON. Os serviços da web JSON usam o termo em seu sentido genérico.

Há algumas diferenças importantes entre SOAP e JSON:

- O conteúdo de uma mensagem SOAP são dados XML, enquanto uma mensagem JSON contém dados JSON. JSON e XML são mecanismos de codificação diferentes para descrever dados estruturados. O JSON tende a ser um mecanismo de codificação mais eficiente, de modo que uma mensagem JSON típica será menor do que a mensagem XML equivalente.
- O JSON é fácil de integrar em aplicativos JavaScript, mas o XML não é. Isso torna o JSON um formato de dados preferencial para muitos desenvolvedores de aplicativos móveis.
- O SOAP fornece um mecanismo para incluir Cabeçalhos em uma mensagem e uma família de especificações para qualidades de serviço (tal como a configuração de segurança e as transações distribuídas). O JSON não fornece esse mecanismo, em vez disso ele depende dos serviços do protocolo de rede HTTP subjacente. Isto resulta em menos opções para proteger e configurar uma carga de trabalho.
- Os serviços da web SOAP são descritos usando documentos WSDL. Os serviços da web JSON são estruturados menos formalmente; eles tendem a ser fracamente acoplados e preferem documentação por exemplo.
- Os serviços da web SOAP têm um formato de erro explícito envolvendo mensagens de Falha de SOAP. Não há equivalente para JSON.

Também há muitas semelhanças entre JSON e SOAP:

- A implementação do CICS de JSON é derivada da arquitetura SOAP e compartilha muitos dos conceitos e artefatos.
- Ambos envolvem programas utilitários off-line que ajudam no mapeamento de dados do aplicativo para a representação de dados externos e a partir dela. Para SOAP há DFHLS2WS e DFHWS2LS, para JSON há DFHLS2JS e DFHJS2LS.
- O mecanismo de implementação para ambas as tecnologias envolve um recurso PIPELINE, um recurso WEBSERVICE e um recurso URIMAP.

esquema JSON

Uma desvantagem de JSON em comparação com SOAP é a dificuldade em documentar a estrutura de uma interface JSON. Os serviços da web SOAP têm a vantagem de documentos WSDL, juntamente com esquemas XML. Um documento WSDL pode não ser fácil de entender, mas há muitas ferramentas disponíveis para trabalhar com documentos WSDL.

O equivalente mais próximo para JSON é a especificação de Esquema JSON disponível em <http://json-schema.org/>. No momento da gravação, ela é uma especificação de rascunho que está criando seu caminho por meio do processo de padronização IETF. O assistente JSON do CICS (DFHLS2JS e DFHJS2LS) fornece uma implementação parcial do rascunho 4 desta especificação emergente. O DFHLS2JS pode ser usado para gerar o esquema JSON e DFHJS2LS pode ser usado para processá-los.

É possível usar o esquema JSON para entender o modelo válido de sintaxe e conteúdo para um serviço da web JSON que foi implementado no CICS. A especificação de esquema JSON não tem o mesmo ecossistema de conjunto de ferramentas que a especificação de esquema XML, mas uma nova geração de ferramentas JSON pode surgir para usar esse formato de dados.

Implementação do CICS para serviços da web baseados em JSON

O CICS suporta dois modos de serviço da web JSON, Solicitação-Resposta e RESTful. O CICS também suporta um cenário programático no qual os aplicativos podem transformar dados JSON em formatos de dados no estilo COBOL sozinhos e vice-versa.

Solicitação-Resposta

O padrão JSON de Solicitação-Resposta é muito semelhante àquele de serviços da web baseados em SOAP no CICS. O serviço da web é implementado usando um PROGRAM no CICS. O PROGRAM possui formatos de dados de entrada e saída, descritos usando estruturas de linguagem (como copybooks COBOL) e o CICS é responsável por transformar mensagens JSON recebidas em dados do aplicativo e por vincular ao aplicativo. O aplicativo retorna dados de saída de volta para o CICS e o CICS transforma isso em dados JSON para retornar ao cliente.

Neste cenário, o cliente JSON deve se conectar ao CICS usando o método HTTP POST.

Um modo de Solicitação-Resposta de serviço da web JSON pode ser desenvolvido em um modo ascendente ou descendente. No modo ascendente, um PROGRAM CICS existente é exposto como um serviço da web JSON usando o Assistente JSON DFHLS2JS. No modo descendente,

um novo serviço da web JSON pode ser desenvolvido para implementar uma interface descrita utilizando esquemas JSON existentes. No modo descendente, o assistente JSON DFHJS2LS é usado para gerar novas estruturas de linguagem e um aplicativo deve ser criado para usá-las.

O padrão de Solicitação-Resposta pode ser usado para construir Serviços da Web JSON que visam a COMMAREA ou os PROGAMS CICS conectados ao canal. Um serviço da web JSON de Solicitação-Resposta pode ser usado somente no modo de provedor (em que o CICS age como o servidor).

RESTful

Este cenário é diferente daquele de serviços da web SOAP. O conceito de um serviço da web JSON RESTful é descrito mais detalhadamente em Conceitos de serviços da web JSON RESTful. Um serviço da web JSON RESTful implementa os princípios de arquitetura do padrão de design Representational State Transfer (REST). É improvável que esse padrão de design seja relevante para aplicativos CICS existentes, portanto, está disponível somente no modo descendente.

Um esquema JSON pode ser processado por DFHJS2LS no modo RESTful. Um aplicativo deve ser gravado para implementar o serviço e ele precisará se comportar de forma diferente, dependendo do método HTTP que foi usado para a solicitação recebida.

O CICS implementa um estilo puro de aplicativo RESTful, em que o formato de dados para POST (criar), GET (consultar) e PUT (substituir) é o mesmo.

Os aplicativos de serviço da web JSON RESTful devem usar uma interface de programa baseada em canal; COMMAREAs não são suportadas. Um serviço da web JSON RESTful pode ser usado somente no modo de provedor (em que CICS age como o servidor).

Modo programático

Neste cenário um aplicativo pode LINK a um programa fornecido pelo CICS, DFHJSON, e solicitar que ele transforme dados do aplicativo em dados JSON, ou dados JSON em dados do aplicativo. Por exemplo, um aplicativo pode usar esse recurso para gerar dados JSON para enviar a um serviço da web JSON remoto. Para fazer isso, ele deve entrar em contato com o serviço da web JSON remoto usando a CICS WEB API.

O CICS não tem suporte integrado para os serviços da web JSON no modo do solicitante, mas um aplicativo pode chamar um serviço da web JSON remoto explorando o modo programático.

Conceitos de serviços da web JSON RESTful

Leia este tópico para entender os conceitos por trás de serviços da web RESTful.

Serviços da web RESTful

REpresentational State Transfer, ou REST, é um padrão de design para interagir com recursos armazenados em um servidor. Cada recurso possui uma identidade, um tipo de dados e suporta um conjunto de ações.

O padrão de design RESTful normalmente é usado em combinação com HTTP, a linguagem da Internet. Neste contexto, a identidade do recurso é seu URI, o tipo de dados é o Tipo de Mídia e as ações são formadas pelos métodos de HTTP padrão (GET, PUT, POST e DELETE).

Esse estilo de serviço difere dos serviços da web no estilo Solicitação-Resposta:

- Serviços de solicitação-resposta iniciam a interação com um Aplicativo, enquanto serviços RESTful geralmente interagem com dados (referidos como 'recursos').
- Os serviços de solicitação-resposta envolvem 'operações' definidas pelo aplicativo, mas os serviços RESTful evitam conceitos específicos do aplicativo.
- Os serviços de solicitação-resposta possuem formatos de dados diferentes para cada mensagem, mas o serviço RESTful geralmente compartilha um formato de dados entre métodos de HTTP diferentes.

Os quatro principais métodos de HTTP definem as quatro operações que são comumente implementadas por Serviços RESTful. O método HTTP POST é usado para criar um recurso, GET é usado para consultá-lo, PUT é usado para mudá-lo e DELETE é usado para destruí-lo. A arquitetura RESTful mais comuns envolve um modelo de dados compartilhados que é usado nessas quatro operações. Esse modelo de dados define a entrada para o método POST (criar), a saída para o método GET (consultar) e a entrada para o método PUT (substituir). Esse padrão de design simples é popular na comunidade RESTful, mas não é o único padrão de design RESTful. O código de status HTTP é usado para indicar o sucesso ou a falha da operação. Algumas APIs RESTful são projetadas de outras maneiras.

Um quinto método de HTTP chamado 'HEAD' às vezes é suportado por serviços da web RESTful. Esse método é equivalente a GET, exceto que ele retorna somente Cabeçalhos HTTP e nenhum dado de Corpo. Às vezes ele é usado para testar a Existência de um recurso. Nem todas as APIs RESTful suportam o uso do método HEAD.

Os aplicativos CICS tradicionais provavelmente não correspondem ao padrão arquitetural RESTful. Aplicativos típicos do CICS implementam várias operações, cada uma das quais terão modelos de dados para formatos de entrada e saída. Essas operações existentes provavelmente não mapeiam diretamente para os quatro métodos de HTTP. Por esse motivo, o padrão arquitetural RESTful é destinado principalmente a novos aplicativos no CICS. Para expor aplicativos do CICS existentes como Serviços RESTful, pode ser necessário agrupá-los com uma nova interface que esteja em conformidade com os princípios RESTful.

O URI

A identidade de um serviço RESTful é indicada por seu URI. Um URI pode ser formado por vários componentes, incluindo o nome do host, o número da porta, o caminho e uma sequência de consultas opcional. O nome de domínio e o número da porta juntos direcionam para um recurso TCPIPService no CICS. Para obter mais informações, consulte Recursos TCPIPService. O caminho de URI é um qualificador e pode ser suficiente para identificar o serviço exclusivamente. No entanto, muitos serviços da web RESTful usam uma sequência de consultas adicional para identificar o recurso exato. Considere os seguintes exemplos:

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

No primeiro exemplo, o caminho de URI é `JSONServices/AccountService`. No segundo exemplo, o caminho é `JSONServices` e há uma sequência de consultas

adicional de Service=Account. Ambos os estilos de URI são considerados aceitáveis para JSON. Esta é uma diferença importante em comparação com SOAP. Sob SOAP o primeiro estilo de URI é preferencial.

O CICS usa um recurso URIMAP para identificar o WEBSERVICE e o PIPELINE apropriados para uso ao processar uma mensagem de entrada. O URIMAP suporta o uso de uma sequência de consultas como parte do atributo path. Portanto, o URIMAP é adequado para uso com ambos os tipos de URI.

Planejando usar serviços da web JSON

Antes de poder planejar usar serviços da web JSON no CICS, é necessário considerar essas questões para cada aplicativo.

Antes de Iniciar

Você planeja usar seus programas de aplicativo existentes ou gravar novos?

Se seus aplicativos existentes foram projetados com uma interface bem definida para a lógica de negócios, você provavelmente poderá usá-los em uma configuração de serviços da web, como um provedor de serviços ou um solicitante de serviço. No entanto, na maioria dos casos, você precisará gravar um programa wrapper que conecta sua lógica de negócios à lógica de serviços da web.

Se planeja gravar novos aplicativos, você deverá tentar manter sua lógica de negócios separada de sua lógica de serviços da web e, mais uma vez, você precisará gravar um programa wrapper para fornecer esta separação. No entanto, se seu aplicativo for projetado com serviços da web em mente, o wrapper poderá ser mais simples de gravar.

Você pretende usar o assistente do CICS para gerar os mapeamentos entre suas estruturas de dados e esquemas JSON?

O assistente fornece uma implementação rápida de muitos aplicativos em uma configuração de serviços da web JSON com pouca ou nenhuma programação adicional. E quando a programação adicional é necessária, ela geralmente é direta e pode ser feita sem alterar a lógica de negócios existente.

No entanto, há casos que são melhor manipulados sem usar o assistente JSON. Por exemplo, se você tiver código existente que mapeia as estruturas de dados para mensagens JSON, não há vantagem na reengenharia de seu aplicativo com o assistente JSON.

Embora o assistente do CICS suporte tipos de dados e estruturas mais comuns, há alguns que não são suportados. Nesta situação, você deve verificar a lista de tipos de dados e estruturas não suportados para o idioma em questão e considerar o fornecimento de uma camada de programa que mapeia seus dados do aplicativo para um formato que o assistente possa suportar. Se isso não for possível, você mesmo precisará analisar a mensagem. Para obter detalhes sobre o que o assistente pode e não pode suportar, consulte Linguagem de alto nível e mapeamento de esquema JSON.

Planejando um aplicativo do provedor de serviços JSON

Em geral, os aplicativos CICS devem ser estruturados para assegurar a separação da lógica de negócios e da lógica de comunicações. Seguir esta prática ajudará a implementar aplicativos novos e existentes em um provedor de serviço da web de

uma maneira direta. Em algumas situações, você precisará interpor um programa wrapper simples entre seu programa de aplicativo e o suporte de serviço da web do CICS.

Figura 12 mostra um aplicativo típico que é particionado para assegurar uma separação entre a lógica de comunicação e a lógica de negócios.

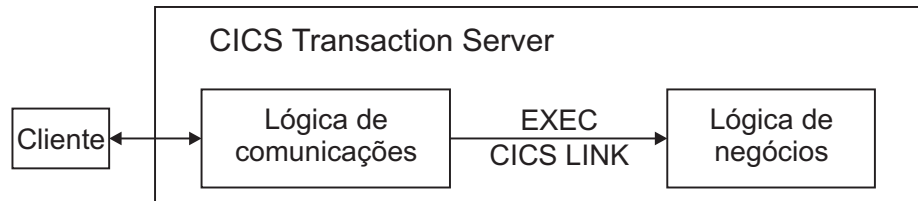


Figura 12. Aplicativo Particionado na Lógica de Comunicações e de Negócios

Em muitos casos, é possível implementar a lógica de negócios diretamente como um aplicativo do provedor de serviços. Isso é ilustrado na Figura 13.

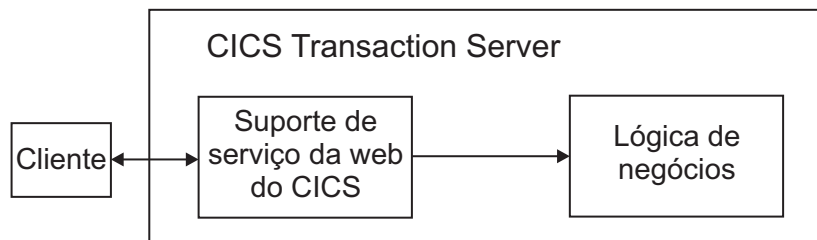


Figura 13. Implementação Simples do Aplicativo CICS como um Provedor de Serviço da Web

Para utilizar esse modelo simples, as seguintes condições se aplicam:

Quando você estiver usando o assistente do CICS para gerar o mapeamento entre o esquema JSON e as estruturas de dados do aplicativo:

Os tipos de dados usados na interface com a lógica de negócios devem ser suportados pelo assistente do CICS. Se este não for o caso, você deverá interpor um programa wrapper entre o suporte de serviço da web do CICS e sua lógica de negócios.

Você também precisará de um programa wrapper ao implementar um programa existente para fornecer um serviço que está em conformidade com uma descrição de serviços da web existente: se você processar a descrição de serviços da web usando o assistente, as estruturas de dados resultantes serão muito improváveis de corresponderem à interface para sua lógica de negócios.

Quando você não estiver usando o assistente do CICS:

Os manipuladores de mensagens no pipeline do provedor de serviços deve interagir diretamente com sua lógica de negócios.

Usando um Programa Wrapper

Use um programa wrapper quando o assistente do CICS não pode gerar código para interagir diretamente com a lógica de negócios. Por exemplo, a interface com a lógica de negócios pode usar uma estrutura de dados que o assistente do CICS não pode mapear diretamente para uma mensagem JSON. Nesta situação, você pode utilizar um programa wrapper para fornecer qualquer manipulação de dados adicionais que seja necessária:

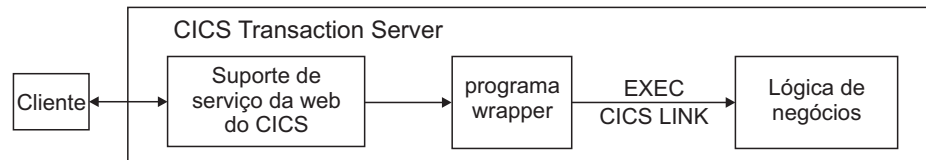


Figura 14. Implementação do Aplicativo CICS como um Provedor de Serviços da Web Utilizando um Programa Wrapper

Será necessário projetar uma segunda estrutura de dados que o assistente pode suportar e utilizar isso como a interface para o programa wrapper. O programa wrapper, então, possui duas funções simples para execução:

- mover dados entre as duas estruturas de dados
- chamar a lógica de negócios utilizando sua interface existente

Manipulação de erros

Se você estiver planejando usar o assistente do CICS, também deverá considerar como manipular a reversão de mudanças quando ocorrerem erros. Quando uma mensagem de solicitação JSON é recebida de um solicitante de serviço, a mensagem JSON é transformada pelo CICS logo antes de ser transmitida ao seu programa de aplicativo. Se ocorrer um erro durante essa transformação, o CICS não recuperará automaticamente nenhum trabalho que foi executado na mensagem. Por exemplo, se você planeja incluir algum processamento adicional na mensagem JSON utilizando manipuladores no pipeline, precisará decidir se eles devem reverter quaisquer mudanças recuperáveis que eles já executaram.

Em mensagens JSON de saída, por exemplo quando seu programa de aplicativo do provedor de serviço está enviando uma mensagem de resposta a um solicitante de serviço, se o CICS encontra um erro ao gerar a mensagem JSON de resposta, todas as mudanças recuperáveis feitas pelo programa de aplicativo são recuperadas automaticamente. Você deve considerar se a inclusão de pontos de sincronização é apropriada para seu programa de aplicativo.

Planejando um aplicativo do solicitante de serviços JSON

O CICS não fornece suporte integrado para os serviços da web JSON no modo do solicitante. Se você desejar chamar outros serviços da web JSON a partir de seu aplicativo CICS, deverá usar a interface vinculável e os comandos de API **EXEC CICS WEB**.

Para obter mais informações sobre como gravar seu próprio solicitante de serviço usando a interface vinculável, consulte Criando um aplicativo cliente do serviço da web JSON.

CICS e z/OS Connect

O z/OS Connect permite que os programas CICS sejam chamados com uma interface JavaScript Object Notation (JSON). O z/OS Connect for CICS 1.0 foi introduzido primeiro como uma alternativa aos recursos JSON do Java Pipelines for JSON que eram fornecidos no CICS TS Feature Pack for Mobile Extensions e integrados no CICS TS Versão 5.2. Desde então, o z/OS Connect cresceu e se tornou um produto separado, chamado z/OS Connect Enterprise Edition, com recursos adicionais. Esta seção descreve as diferenças entre as ofertas alternativas e as etapas que levam à implementação.

Para obter uma visão geral de serviços JSON, consulte "CICS como um provedor de serviços para solicitações JSON no IBM Knowledge Center.

Considerações sobre manutenção para z/OS Connect Enterprise Edition

Como o z/OS Connect Enterprise Edition pode ser executado independente ou dentro do Liberty integrado no CICS, os dois ambientes devem ser descritos exclusivamente. É possível executar ambos os ambientes simultaneamente, mas eles não são mutuamente exclusivos, portanto, é necessário ter cuidado ao aplicar a manutenção nas bibliotecas do cliente WebSphere Liberty Profile (WLP) que contém os módulos BBOA*.

Para obter mais informações, consulte **Mantendo a manutenção do CICS TS 5.3 e z/OS Connect EE 2.0 em sincronização**.

z/OS Connect Enterprise Edition

O z/OS Connect Enterprise Edition é um produto IBM solicitado separadamente. Ele é construído sobre os recursos do z/OS Connect for CICS 1.0, o qual incluía o suporte para serviços JSON. O z/OS Connect Enterprise Edition permite que os desenvolvedores de API construam APIs JSON a partir de serviços JSON. As APIs são construídas e empacotadas com o Editor de API baseado em Eclipse que é fornecido com o z/OS Connect Enterprise Edition, em seguida, implementados no tempo de execução do z/OS Connect. O pacote de API inclui definições do Swagger 2.0 para tornar mais fácil para os desenvolvedores incorporar as APIs em seus aplicativos. Recursos-chave do z/OS Connect, tal como a verificação de segurança da autorização para chamada de serviço, a criação de registros System Management Facility (SMF) e a criação de log de solicitações de serviço RESTful também se aplicam às APIs.

O z/OS Connect Enterprise Edition pode ser configurado para execução em um servidor Liberty no CICS. Essa configuração permite a conectividade local de alto desempenho para programas CICS.

Os serviços JSON que foram desenvolvidos para uso com o z/OS Connect for CICS 1.0, e a maioria dos serviços JSON que foram desenvolvidos para uso com o Java Pipelines for JSON, podem ser hospedados no z/OS Connect Enterprise Edition. Consulte o "z/OS Connect for CICS 1.0" na página 36, para obter detalhes adicionais. No entanto, os usuários do Editor de API que é fornecido com o z/OS Connect Enterprise Edition devem estar cientes de algumas restrições. Consulte o Usando as APIs do z/OS Connect Enterprise Edition, para obter detalhes adicionais.

Para obter mais informações sobre z/OS Connect Enterprise Edition, consulte Documentação do z/OS Connect Enterprise Edition no IBM Knowledge Center.

z/OS Connect for CICS 1.0

O z/OS Connect for CICS 1.0 é um recurso gratuito fornecido no CICS TS a partir da Versão 5.2. Ele é amplamente equivalente no Java Pipelines for JSON e a maioria dos serviços da web JSON pode ser reimplementada de um ambiente para o outro sem mudanças nos aplicativos ou arquivos WSBind. No entanto, o URI e a configuração de segurança podem ser diferentes em cada ambiente. Os serviços da web JSON que são implementados no z/OS Connect for CICS 1.0 também podem ser implementados no z/OS Connect Enterprise Edition.

A versão do z/OS Connect que está integrada no CICS TS compartilha muito da função de z/OS Connect no WebSphere Liberty para z/OS, mas é otimizada para acesso local para o CICS e usa serviços padrão do CICS. Se você está familiarizado com o z/OS Connect no WebSphere Liberty for z/OS, as principais diferenças entre ele e o z/OS Connect for CICS 1.0 são as seguintes:

- O z/OS Connect for CICS 1.0 suporta somente a implementação de serviços da web JSON para CICS. Ele não pode ser usado para acessar outros subsistemas z/OS como IMS ou Lote.
- O z/OS Connect for CICS 1.0 integra-se com recursos do CICS. Os serviços da web JSON são implementados usando recursos WEBSERVICE e URIMAP.
- As técnicas de ativação de serviço JSON do Java Pipelines for JSON são suportadas com o z/OS Connect for CICS 1.0, incluindo aquelas que normalmente não estão disponíveis no z/OS Connect.
- Os serviços JSON podem ser desenvolvidos utilizando tanto estratégias de desenvolvimento descendente quanto crescente.
- Os serviços da web JSON gerados para Java Pipelines for JSON podem ser implementados no z/OS Connect for CICS 1.0.

Para obter mais informações sobre o z/OS Connect no WebSphere Liberty for z/OS, consulte Visão geral do IBM z/OS Connect.

Java Pipelines para JSON (ou o Feature Pack for Mobile Extensions)

O Mobile Extensions Feature Pack 1.0 funcionava com o CICS TS for z/OS Versão 5.1 e Versão 4.2. O Feature Pack fornecia a capacidade de expor aplicativos CICS como serviços da web RESTful com cargas úteis JSON, chamar os aplicativos JSON existentes e converter JSON para e a partir dos dados do aplicativo. A partir da Versão 5.2, esse recurso foi integrado no CICS TS.

Comparando os recursos do z/OS Connect

Esta seção compara os recursos do z/OS Connect Enterprise Edition e z/OS Connect for CICS 1.0. Uma comparação para o Java Pipelines for JSON (ou o Feature Pack for Mobile Extensions) também é incluída porque ela precedeu o z/OS Connect for CICS 1.0.

Tabela 1. Comparando os recursos do z/OS Connect Enterprise Edition, do z/OS Connect for CICS 1.0 e dos Java Pipelines for JSON (ou o Feature Pack for Mobile Extensions)

Capacidade	z/OS Connect Enterprise Edition	z/OS Connect for CICS 1.0	Java Pipelines for JSON (ou o Feature Pack for Mobile Extensions)
Suporte para recursos padrão do z/OS Connect	Sim. A Descoberta de Serviço requer configuração extra.	Sim. A Descoberta de Serviço requer configuração extra.	Não
Ativar programas CICS como serviços da web JSON	Sim	Sim	Sim
Suporte para serviços RESTful JSON	Sim	Sim	Sim
Suporte para APIs RESTful	Sim	Não	Não
Suporte para iniciar serviços JSON remotos	Não. Use DFHJSON.	Não. Use DFHJSON.	Não (use DFHJSON)
Suporte para transformação de dados do CICS que usa os assistentes DFHLS2JS e DFHJ2LS	Para serviços, sim. Para APIs, use os assistentes fornecidos pelo z/OS Connect EE	Sim	Sim
Suporte para especificação de esquema JSON	Sim (draft-04)	Sim (draft-04)	Sim (draft-04)
Opção de tecnologias analisadoras para transformação JSON	Sim	Sim	Não
Suporte para programas de manipulador PIPELINE	Não	Não	Sim
Suporte para programas do interceptor	Sim, usando interceptores z/OS Connect	Sim, usando interceptores z/OS Connect	Sim, usando manipuladores Axis2
Acesso ao aplicativo para contêineres de controle CICS	Sim	Sim	Sim
Implementação de WSBIND por meio de um recurso CICS PIPELINE	Para serviços, sim. A implementação de APIs envolve artefatos adicionais (consulte a documentação do z/OS Connect EE no IBM Knowledge Center para obter detalhes).	Sim	Sim
Configuração com recursos URIMAP e WEBSERVICE	Para serviços, sim	Sim	Sim

Tabela 1. Comparando os recursos do z/OS Connect Enterprise Edition, do z/OS Connect for CICS 1.0 e dos Java Pipelines for JSON (ou o Feature Pack for Mobile Extensions) (continuação)

Capacidade	z/OS Connect Enterprise Edition	z/OS Connect for CICS 1.0	Java Pipelines for JSON (ou o Feature Pack for Mobile Extensions)
Configuração de rede	Usando o WebSphere Liberty	Usando o WebSphere Liberty	Usando um recurso TCPIP SERVICE
Configuração de segurança	Usando a segurança do WebSphere Liberty e do CICS	Usando a segurança do WebSphere Liberty e do CICS	Usando CICS
Suporte para Segurança da Camada de Transporte (TLS)	Sim	Sim	Sim
Estatísticas, monitoramento, saídas do usuário e outra infraestrutura de diagnósticos	Consistente com o WebSphere Liberty no CICS	Consistente com o WebSphere Liberty no CICS	Consistente com Pipelines no CICS
Ambiente da JVM	Um servidor Liberty JVM que é preferivelmente isolado para uso exclusivo pelo z/OS Connect	Um servidor Liberty JVM que é preferivelmente isolado para uso exclusivo pelo z/OS Connect	Um servidor JVM não OSGi configurado com JAVA_PIPELINE=YES , preferencialmente isolado para uso exclusivo pelo Java Pipelines for JSON

Capítulo 2. Configurando Serviços da Web no CICS

É possível configurar o CICS para suportar serviços da web, no qual aplicativos CICS podem se tornar solicitantes de serviço da web ou provedores de serviços. O CICS suporta diferentes especificações de serviço da web, incluindo anexos binários e endereçamento de serviços da web. Também é possível configurar o CICS para aceitar solicitações de serviço da web do WebSphere MQ ou HTTP e recuperar arquivos WSDL do WSRR.

Configurando seu Sistema CICS para Serviços da Web

Antes de poder usar serviços da web, seu sistema CICS deve ser configurado corretamente.

Procedimento

1. Assegure que você tenha instalado o suporte Ambiente de Linguagem para PL/I. Para obter mais informações, consulte Instalando o Suporte ao Ambiente de Linguagem.
2. Ative o Suporte ao z/OS para Unicode. Você deve ativar os serviços de conversão do z/OS e instalar uma imagem de conversão que especifica as conversões de dados que deseja que o CICS execute entre mensagens SOAP e um programa de aplicativo. Para obter mais informações, consulte z/OS Unicode Services User's Guide and Reference.

Recursos do CICS para serviços da web

Recursos PIPELINE, WEBSERVICE, URIMAP e TCPIPSERVICE Suportam Serviços da Web no CICS.

PIPELINE

Uma definição de recurso PIPELINE é necessária para cada serviço da Web. Ele fornece informações sobre os programas manipuladores de mensagem que agem em uma solicitação de serviço e na resposta. Geralmente, uma única definição de recurso PIPELINE define uma infraestrutura que pode ser usada por muitos aplicativos. As informações sobre os manipuladores de mensagem são fornecidas indiretamente: a definição de recurso PIPELINE especifica o nome de um arquivo z/OS UNIX que contém uma descrição de XML dos manipuladores e suas configurações.

Um recurso PIPELINE criado para um solicitante de serviços não pode ser utilizado por um fornecedor de serviços e vice-versa. Os dois tipos de definições de PIPELINE são diferenciados pelo conteúdo do arquivo de configuração de pipeline que está especificado no atributo CONFIGFILE: para um provedor de serviços, o elemento de nível superior é `<provider_pipeline>`; para um solicitante de serviços, é `<requester_pipeline>`.

WEBSERVICE

Uma definição de recurso WEBSERVICE é necessária somente quando o mapeamento entre a estrutura de dados do aplicativo e as mensagens SOAP foi gerado usando o assistente de serviços da web do CICS. Ele define aspectos do ambiente de tempo de execução para um programa de aplicativo CICS implementado em uma configuração de serviços da web.

Embora o CICS forneça os mecanismos de definição de recurso usuais para recursos WEBSERVICE, eles geralmente são criados automaticamente a partir de um arquivo de ligação de serviço da web quando o diretório de recebimento para a definição de recurso PIPELINE é varrido. Isso pode ocorrer quando o recurso PIPELINE é instalado ou como resultado de um comando PERFORM PIPELINE SCAN. Os atributos aplicados ao recurso WEBSERVICE neste caso são fornecidos a partir de um arquivo de ligação de serviços da Web, que é criado pelo assistente de serviços da Web; as informações no arquivo de ligação vêm da descrição do serviço da Web ou são fornecidas como um parâmetro do assistente de serviços da web.

Um recurso WEBSERVICE criado para um solicitante de serviços não pode ser utilizado por um fornecedor de serviços e vice-versa. Os dois tipos de recurso WEBSERVICE são diferenciados pelo atributo PROGRAM na definição de recurso: para um provedor de serviços, o atributo deve ser especificado; para um solicitante de serviços, ele deve ser omitido.

URIMAP

Uma definição de URIMAP é necessária em um provedor de serviços quando ele contém informações que mapeiam o URI de uma solicitação de serviço da web de entrada para os outros recursos (como o recurso PIPELINE) que atenderão a solicitação. Essa definição de URIMAP também será necessária se você estiver utilizando a autenticação básica HTTP, porque a definição de recurso URIMAP especifica que as informações de ID do usuário do solicitante de serviços são transmitidas em um cabeçalho de autorização HTTP para o provedor de serviços.

Uma segunda definição de URIMAP opcional pode existir em um provedor de serviços para descoberta de WSDL. Esta definição de recurso URIMAP contém informações que mapeiam o URI de uma solicitação de entrada para o documento ou documentos WSDL associados ao serviço da web.

Para provedores de serviços implementados utilizando o assistente de serviços da web do CICS, embora o CICS forneça os mecanismos de definição de recurso usuais, os recursos URIMAP geralmente são criados automaticamente quando o diretório de recebimento é varrido. Esta varredura ocorre quando o recurso PIPELINE é instalado ou como resultado de um comando PERFORM PIPELINE SCAN. O recurso URIMAP que fornece ao CICS as informações para associar o recurso WEBSERVICE a um URI específico é um recurso necessário. Os atributos para este recurso são especificados por um arquivo de ligação de serviço da web no diretório de recebimento. O recurso URIMAP que fornece ao CICS as informações para associar o archive WSDL ou documento WSDL a um URI específico é um recurso opcional e é criado se um arquivo WSDL ou archive WSDL está presente no diretório de recebimento. Para obter mais informações sobre a criação de recursos URIMAP para provedores de serviços da web, consulte Criando um Provedor de Serviço da Web Usando o Assistente de Serviços da Web.

Para solicitantes de serviços, o CICS não cria nenhum recurso URIMAP automaticamente quando o recurso PIPELINE é instalado ou como resultado de um comando PERFORM PIPELINE SCAN. Os solicitantes de serviço não precisam usar recursos URIMAP quando eles fazem solicitações; eles podem especificar o URI da solicitação de saída diretamente no programa de aplicativo. No entanto, se você criar um recurso URIMAP para a solicitação do cliente, e seus solicitantes de serviço usarem o recurso URIMAP para fornecer o URI, terá estas vantagens:

- Os administradores do sistema podem gerenciar mudanças no terminal da conexão, portanto, não é necessário recompilar seus aplicativos se o URI de um provedor de serviços for alterado.
- É possível optar por fazer o CICS manter abertas as conexões que foram abertas com o recurso URIMAP após o uso, e colocá-las em um conjunto para reutilização pelo aplicativo para solicitações subsequentes, ou por outro aplicativo que chama o mesmo serviço. O conjunto de conexões ficará disponível apenas quando você especificar um recurso URIMAP que tem o atributo SOCKETCLOSE configurado. Para obter mais informações sobre os benefícios de desempenho do conjunto de conexões, consulte Conjunto de Conexões para Desempenho de Cliente HTTP.

Configurar atributos de recurso URIMAP de uma determinada maneira pode permitir que solicitações de entrada sejam processadas por transações do usuário conectadas diretamente e efetuem bypass da tarefa de conexão da web. Para obter mais informações, consulte As solicitações de HTTP HTTP são processadas por transações do usuário diretamente anexadas.

TCPIPSERVICE

Uma definição de TCPIPSERVICE é necessária em um provedor de serviços que usa o transporte HTTP. Ela contém informações sobre a porta na qual solicitações de entrada são recebidas.

Os recursos que são necessários para suportar um programa de aplicativo específico dependem dos critérios a seguir:

- Se o programa de aplicativo é um provedor de serviços ou um solicitante de serviço.
- Se o aplicativo é implementado com o assistente de serviços da web do CICS.

Solicitante ou Provedor de Serviço	Assistente de Serviços da Web do CICS Usado	PIPELINE requerido	WEBSERVICE requerido	URIMAP requerido	TCPIPSERVICE requerido
Provider	Sim	Sim	Sim (mas consulte nota 1)	Sim (mas consulte nota 1)	Consulte nota 2
Provider	Não	Sim	Não	Sim	Consulte nota 2
Solicitador	Sim	Sim	Sim	Consulte nota 3	Não
Solicitador	Não	Sim	Não	3	Não

Notas:

1. Quando o assistente de serviços da web do CICS é usado para implementar um programa de aplicativo, o WEBSERVICE e dois recursos URIMAP podem ser criados automaticamente quando o diretório de recebimento do PIPELINE é varrido. O primeiro recurso URIMAP é necessário e fornece ao CICS as informações para associar o recurso WEBSERVICE a um URI específico. O segundo recurso URIMAP é opcional e fornece ao CICS as informações para associar o archive WSDL ou o documento WSDL a um URI específico para que os solicitantes externos possam usar o URI para descobrir o archive WSDL ou o documento WSDL. O diretório de recebimento da varredura de PIPELINE ocorre quando o recurso PIPELINE é instalado ou como resultado de um comando PERFORM PIPELINE SCAN.

2. Um recurso TCPIPSERVICE é necessário quando o transporte HTTP é usado. Quando o transporte WebSphere MQ é usado, um recurso TCPIPSERVICE não é necessário.
3. Um recurso URIMAP é opcional para um solicitante de serviços e o assistente de serviços da web do CICS não gera um automaticamente. Ao definir seus próprios recursos URIMAP para solicitantes de serviço usarem, é possível implementar definição do conjunto de conexões e gerenciar mudanças nos URIs para provedores de serviços.

Configurar atributos de recurso TCPIP de uma determinada maneira pode permitir que solicitações de entrada sejam processadas por transações do usuário conectadas diretamente, efetuando bypass da tarefa de conexão da web. Para obter mais informações, consulte As solicitações de HTTP HTTP são processadas por transações do usuário diretamente anexadas.

Geralmente, quando você implementa muitos aplicativos de serviços da web em um sistema CICS, você tem mais de um de cada tipo de recurso. Nesse caso, é possível compartilhar alguns recursos entre os aplicativos. Cada arquivo ou recurso de serviços da web é associado a um ou mais recursos do CICS de outros tipos.

Tabela 2. Outros recursos do CICS que estão associados a cada arquivo e recurso de serviços da web

Arquivo ou recurso de serviços da web	Recursos Associados
Arquivo de configuração do pipeline	<ul style="list-style-type: none"> Mais de um recurso PIPELINE que se refere ao arquivo.
PIPELINE	<ul style="list-style-type: none"> Mais de um recurso URIMAP que se refere ao recurso PIPELINE. Mais de um recurso WEBSERVICE que se refere ao recurso PIPELINE. Mais de um arquivo de ligação de serviço da web no diretório de recebimento do recurso PIPELINE.
Arquivo de Ligação de Serviços da Web	<ul style="list-style-type: none"> Um recurso URIMAP que é gerado automaticamente a partir do arquivo de ligação. É possível definir recursos URIMAP adicionais para um provedor de serviços e você pode definir recursos URIMAP para um solicitante de serviço. Um recurso WEBSERVICE que é gerado automaticamente a partir do arquivo de ligação. É possível definir recursos WEBSERVICE adicionais se precisar.
WEBSERVICE	<ul style="list-style-type: none"> Mais de um recurso URIMAP. Se o recurso WEBSERVICE é gerado automaticamente a partir do arquivo de ligação para um provedor de serviços, o CICS gera um recurso URIMAP correspondente. É possível definir recursos URIMAP adicionais para um provedor de serviços e você pode definir recursos URIMAP para um solicitante de serviço.

Tabela 2. Outros recursos do CICS que estão associados a cada arquivo e recurso de serviços da web (continuação)

Arquivo ou recurso de serviços da web	Recursos Associados
URIMAP	<ul style="list-style-type: none"> Apenas um recurso TCPIService quando ele é nomeado explicitamente no recurso URIMAP.
TCPIService	<ul style="list-style-type: none"> Muitos recursos URIMAP.

Descoberta de Serviços da Web

Os documentos WSDL associados a um serviço da web de modo de Provedor são automaticamente publicados na web.

Existe uma convenção entre os ambientes de hospedagem de serviço da web que permitem que o WSDL para um serviço da web seja consultado por um cliente remoto (geralmente um Desenvolvedor de Aplicativo que usa um navegador da web) usando a URI para o serviço da web com sufixo de `?wsdl`. Essa convenção pode facilitar a distribuição do WSDL para as partes interessadas sem a necessidade de um repositório WSDL formal. Essa convenção está implementada no CICS.

Por exemplo, você pode ter um serviço da web hospedado no CICS e publicado na URI a seguir:

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

O documento WSDL associado poderia ser recuperado solicitando a URI a seguir usando um navegador da web:

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

Os documentos WSDL para os provedores de serviço podem ser publicados para descoberta usando os recursos URIMAP. Ao instalar cada recurso PIPELINE, o CICS varre o diretório especificado no atributo WSDIR do recurso PIPELINE (o diretório de recebimento). Se este diretório contiver um archive WSDL ou documento WSDL, um segundo recurso URIMAP será instalado. Este novo recurso URIMAP fornece ao CICS as informações para associar o archive WSDL ou um documento WSDL a uma URI específica para que os solicitantes possam usar a URI para descobrir o archive WSDL ou documento WSDL. Esta URI tem o mesmo caminho que a URI associada ao WEBSERVICE ao sufixo `?wsdl` anexado.

O archive WSDL pode conter um ou mais documentos WSDL. Se o diretório de recebimento contiver um archive WSDL e um documento WSDL, a URI retornará somente o archive WSDL. O formato de archive que é suportado é o tipo de arquivo `.zip`. Também é possível descobrir o archive WSDL ou o documento WSDL usando SPI e CEMT. O documento WSDL em um archive WSDL pode ser usado para validação de mensagem SOAP.

Configurando o CICS para Usar o Transporte do WebSphere MQ

Para usar o transporte do WebSphere MQ com os serviços da web SOAP no CICS, deve-se configurar sua região CICS de forma apropriada.

Sobre Esta Tarefa

Nota: Não é possível usar o transporte do WebSphere MQ para serviços da web JSON.

Procedimento

1. Inclua a biblioteca do WebSphere MQ, *thlqual.SCSQAUTH*, na concatenação de STEPLIB em seu procedimento do CICS. Inclua a biblioteca após as bibliotecas do CICS para assegurar que o código correto seja usado. *thlqual* é o qualificador de alto nível para as bibliotecas do WebSphere MQ.
2. Inclua as bibliotecas do WebSphere MQ a seguir na concatenação de DFHRPL em seu procedimento do CICS. Inclua as bibliotecas após as bibliotecas do CICS para assegurar que o código correto seja usado.

thlqual.SCSQCICS

thlqual.SCSQLOAD

thlqual.SCSQAUTH

thlqual é o qualificador de alto nível para as bibliotecas do WebSphere MQ. Se você estiver usando a saída cruzada da API do CICS-WebSphere MQ (CSQCAPX), também inclua o nome da biblioteca que contém o módulo de carregamento para o programa. A biblioteca SCSQCICS é necessária somente se você desejar executar amostras fornecidas pelo WebSphere MQ. Caso contrário, ela pode ser removida do procedimento do CICS.

3. Instale um recurso MQCONN para a região do CICS. O recurso MQCONN especifica os atributos da conexão entre o CICS e o WebSphere MQ, incluindo o nome do gerenciador de filas do WebSphere MQ padrão ou do grupo de filas compartilhadas para a conexão. Para obter mais informações, consulte Configurando um recurso MQCONN.
4. Especifique o parâmetro de inicialização do sistema CICS **MQCONN=YES** para iniciar a conexão de CICS-WebSphere MQ automaticamente na inicialização do CICS. Veja detalhes na seção Parâmetro de inicialização do sistema MQCONN.
5. Se você estiver usando o adaptador do CICS-WebSphere MQ em um sistema CICS que possui comunicação inter-regional (IRC) com sistemas CICS remotos, assegure que o recurso IRC esteja OPEN antes de iniciar o adaptador, especificando o parâmetro de inicialização do sistema CICS IRCSTRT=YES. O recurso IRC deve estar OPEN se o método de acesso IRC for definido como memória cruzada; ou seja, ACCESSMETHOD(XM).
6. Assegure que os identificadores de conjunto de caracteres codificados (CCSIDs) usados por seu gerenciador de filas e pelo CICS e que as páginas de códigos UTF-8 e UTF-16 sejam configurados para serviços de conversão do z/OS. A página de códigos do CICS é especificada no parâmetro de inicialização do sistema **LOCALCCSID**.
7. Atualize seu CSD do CICS conforme a seguir:
 - a. Se você não compartilhar seu CSD com liberações anteriores do CICS, remova os grupos CSQCAT1 e CSQCKB e quaisquer cópias desses grupos ou de itens desses grupos de seu CSD. Você também deve excluir o TDQUEUE CKQQ do grupo CSQCAT1. A definição para CKQQ agora é fornecida no grupo do CICS CSD, DFHDCTG.
 - b. Se você compartilhar seu CSD com liberações anteriores do CICS, assegure que CSQCAT1 e CSQCKB, e quaisquer cópias desses grupos ou de seu conteúdo, não estejam instalados para o CICS TS 4.1 ou CICS TS 3.2. Você também deve excluir o CKQQ TDQUEUE do grupo CSQCAT1. A definição para CKQQ agora é fornecida no grupo do CICS CSD, DFHDCTG. Para as

liberações do CICS TS anteriores ao CICS TS 3.2, instale os grupos CSQCAT1 e CSQCKB como parte de uma lista de grupos, após instalar o DFHLIST, para substituir o grupo DFHMQ e instalar corretamente as definições necessárias.

8. Atualize as definições do WebSphere MQ para a fila de mensagens não entregues, a fila de transmissão padrão e os objetos do adaptador do CICS-WebSphere MQ. É possível usar CSQ4INYG de amostra, mas pode ser necessário alterar o nome da fila de inicialização para corresponder ao nome da fila de inicialização padrão na definição de recurso MQINI para sua região do CICS. É possível usar este membro na concatenação do CSQINP2 DD do procedimento de inicialização do gerenciador de filas ou você pode usá-lo como entrada na função COMMAND do utilitário CSQUTIL para emitir os comandos DEFINE necessários. O uso do utilitário CSQUTIL é preferível porque você não precisa redefinir estes objetos toda vez que reinicia o WebSphere MQ.

O Transporte WebSphere MQ

O CICS pode receber e enviar mensagens SOAP ao WebSphere MQ usando o transporte WebSphere MQ, tanto na função de provedor de serviços como na de solicitante de serviço.

Como um **provedor de serviços**, o CICS usa o acionamento do WebSphere MQ para processar mensagens SOAP de uma fila do aplicativo. O acionamento funciona usando uma fila de inicialização e filas locais. Uma definição de fila local (aplicativo) inclui as informações a seguir:

- Os critérios para quando uma mensagem do acionador é gerada. Por exemplo, quando a primeira mensagem chega na fila local ou para cada mensagem que chega na fila local. Para processamento de SOAP do CICS, especifique que o acionamento ocorre quando a primeira mensagem chega na fila local.

A definição de fila local também pode especificar que dados do acionador sejam transmitidos ao aplicativo de destino e, no caso de processamento de SOAP do CICS (transação CPIL), isto especifica a URL de destino padrão a ser usada se isto não for transmitido com a mensagem de entrada.

- O *nome do processo* que identifica a *definição de processo*. A definição de processo descreve como a mensagem é processada. No caso de processamento de SOAP do CICS, especifique a transação CPIL.
- O nome da fila de inicialização para a qual a mensagem do acionador deve ser enviada.

Quando uma mensagem chega na fila local, o Gerenciador de Filas gera e envia uma mensagem do acionador à fila de inicialização especificada. A mensagem do acionador inclui as informações da definição de processo. O monitor acionador recupera a mensagem do acionador da fila de inicialização e planeja a transação CPIL para iniciar o processamento das mensagens na fila local. Para obter mais informações sobre acionamento, consulte .

É possível configurar o CICS, para que quando uma mensagem chegue em uma fila local o monitor acionador (fornecido pelo WebSphere MQ) planeje a transação CPIL para processar as mensagens na fila local e conduza o pipeline CICS para processar as mensagens SOAP na fila.

Quando o CICS constrói uma resposta para uma mensagem SOAP que é recebida do WebSphere MQ, o campo do ID de correlação é preenchido com o ID de mensagem da mensagem de entrada, a menos que a opção de relatório

MQRO_PASS_CORREL_ID tenha sido configurada. Se esta opção de relatório tiver sido configurada, o ID de correlação será propagado da mensagem de entrada para a resposta.

Como um **solicitante de serviço**, em solicitações de saída é possível especificar que as respostas para o serviço da web de destino são retornadas em uma fila de resposta específica.

Em ambos os casos, o CICS e WebSphere MQ requerem configuração para definir os recursos e filas necessários.

Definindo Filas Locais em um Provedor de Serviços

Para usar o transporte do WebSphere MQ em um provedor de serviços, você deve definir uma ou mais filas locais que armazenam mensagens de solicitação até elas serem processadas e um processo acionador que especifica a transação do CICS que processará as mensagens de solicitação.

Procedimento

1. Defina uma fila de inicialização. Utilize o comando a seguir:

```
DEFINE
QLOCAL('initiation_queue')
DESCR('description')
```

em que *initiation_queue* é o mesmo que o valor especificado para o atributo QNAME da definição de recurso MQMONITOR instalada para a região CICS ou o valor especificado para o atributo INITQNAME da definição de recurso MQCONN instalada.

2. Para cada fila de solicitações local, defina um objeto QLOCAL. Utilize o comando a seguir:

```
DEFINE
QLOCAL('queueename')
DESCR('description')
PROCESS(processname)
INITQ('initiation_queue')
TRIGGER
TRIGTYPE(FIRST)
TRIGDATA('default_target_service')
BOTHRESH(nnn)
BOQNAME('requeueename')
```

em que:

- *queueename* é o nome da fila local.
- *processname* é o nome da instância de processo que identifica o aplicativo iniciado pelo gerenciador de filas quando um evento acionador ocorre. Especifique o mesmo nome em cada objeto QLOCAL.
- *initiation_queue* é o nome da fila de inicialização a ser usada; por exemplo, a fila de inicialização especificada no atributo QNAME da definição de recurso MQMONITOR instalada para a região CICS.
- *default_target_service* é o serviço de destino padrão a ser usado se um serviço não é especificado na solicitação. O serviço de destino é no formato '/string' e é usado para corresponder ao caminho de uma definição de URIMAP; por exemplo, '/SOAP/test/test1'. O primeiro caractere deve ser '/'.

- *nnn* é o número de novas tentativas que são tentadas.
 - *requeuename* é o nome da fila para a qual mensagens com falha são enviadas.
3. Defina um objeto PROCESS que especifica o processo acionador. Utilize o comando a seguir:

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

em que:

processname é o nome do processo e deve ser igual ao nome que é usado ao definir as filas de solicitações.

Definindo Filas Locais em um Solicitante de Serviço

Quando você usa o transporte do WebSphere MQ para solicitações de saída em um solicitante de serviço, é possível especificar no URI para o serviço da web de destino que suas respostas devem ser retornadas em uma fila de resposta predefinida. Se fizer isso, você deverá definir cada fila de resposta com um objeto QLOCAL.

Sobre Esta Tarefa

Se o URI associado a uma solicitação não especificar uma fila de resposta, o CICS usará uma fila dinâmica para a resposta.

Procedimento

Opcional: Para definir cada objeto QLOCAL que especifica uma fila de resposta predefinida, use o comando a seguir.

```
DEFINE  
QLOCAL('reply_queue')  
DESCR('description')  
BOTHRESH(nnn)
```

em que:

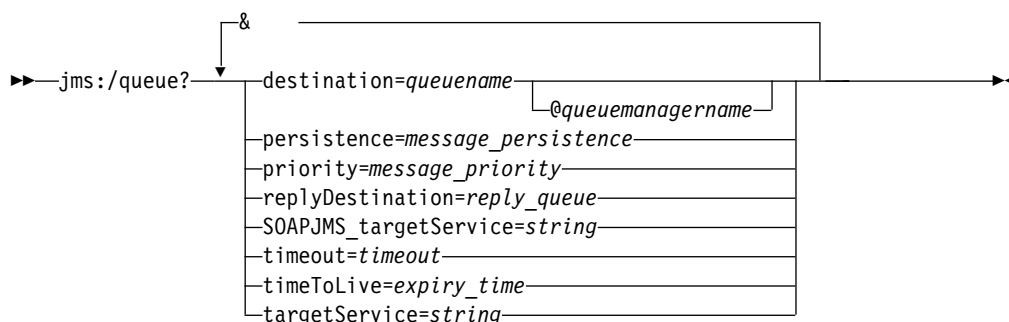
reply_queue é o nome da fila local.

nnn é o número de novas tentativas que serão tentadas.

O URI para o transporte do IBM MQ

Quando a comunicação entre o solicitante de serviço e o provedor de serviços usa IBM MQ, o URI do destino está em um formato que identifica o destino como uma fila e inclui informações para especificar como a solicitação e a resposta devem ser manipuladas pelo IBM MQ.

Sintaxe



Opções

O CICS usa as opções a seguir; outros provedores de serviço da web podem usar opções adicionais que não são descritas aqui. O URI inteiro é transmitido ao provedor de serviços, mas o CICS ignora qualquer opção que ele não suporte e que esteja codificada no URI. O CICS não faz distinção entre maiúsculas e minúsculas nos nomes de opção. Entretanto, algumas outras implementações que suportam este estilo de URI fazem distinção entre maiúsculas e minúsculas.

destination=queueename [*@queueanagername*]

queueename é o nome da fila de entrada no gerenciador de fila de destino

queueanagername é o nome do gerenciador de fila de destino

persistence=message_persistence

Especifique uma das opções seguintes:

- 0** A persistência é definida pela persistência de fila padrão.
- 1** As mensagens não são persistentes.
- 2** As mensagens são persistentes.

Se a opção não for especificada ou for especificado incorretamente, a persistência da fila padrão será usada.

priority=message_priority

Especifica a prioridade da mensagem. O CICS suporta valores de número inteiro no intervalo 0 a 9 para prioridades da mensagem, em que 9 é designado para as mensagens de prioridade mais alta e 0 é designado para a mensagens de prioridade mais baixa. Como alternativa, especifique **-1** para usar a prioridade padrão que é definida para a fila de destino.

replyDestination=reply_queue

Especifica a fila a ser usada para a mensagem de resposta. Se essa opção não for especificada, o CICS usará uma fila dinâmica para a mensagem de resposta. Você deve definir a fila de resposta em um objeto QLOCAL antes de usar esta opção.

SOAPJMS_targetService=string

Identifica o serviço de destino. Se o CICS for o provedor de serviços, o serviço de destino deverá estar no formato `'/string'`, pois o CICS usa isto como o caminho ao tentar corresponder ao URIMAP. Se essa opção não for especificada, o valor especificado em TRIGDATA na fila de entrada no provedor de serviços será utilizado.

timeout=timeout

O tempo limite em milissegundos para o qual o solicitante de serviço aguarda

por uma resposta. Se um valor igual a zero for especificado, ou se esta opção for omitida, a solicitação não atingirá tempo limite.

timeToLive=expiry-time

Especifica o horário de validação para a solicitação em milissegundos. Se a opção não for especificado ou for especificada incorretamente, a solicitação não expirará.

targetService=string

Identifica o serviço de destino. Se o CICS for o provedor de serviços, o serviço de destino deverá estar no formato '/string', pois o CICS usa isto como o caminho ao tentar corresponder ao URIMAP. Se essa opção não for especificada, o valor especificado em TRIGDATA na fila de entrada no provedor de serviços será utilizado.

Exemplo

Este exemplo mostra um URI para o transporte de IBM MQ:

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

Para obter informações sobre "connectionFactory" e "initialContextFactory", consulte a documentação do produto *IBM MQ*.

Configurando CICS para Suportar Mensagens Persistentes

O CICS fornece suporte para enviar mensagens persistentes usando o protocolo de transporte do WebSphere MQ para um aplicativo do provedor de serviço da web que é implementado em uma região do CICS.

Sobre Esta Tarefa

O CICS usa o Business Transaction Services (BTS) para assegurar que mensagens persistentes sejam recuperadas no evento de uma falha do sistema CICS. Para que ele funcione corretamente, siga estas etapas:

Procedimento

1. Use o IDCAMS para definir a fila de solicitações local e o arquivo de repositório como MVS. Você deve especificar um valor adequado para STRINGS para a definição de arquivo. O valor padrão 1 é improvável que seja suficiente e é recomendado que você use 10 em substituição.
2. Defina a fila de solicitações local e o arquivo de repositório para o CICS. Detalhes de como definir a fila de solicitações local para o CICS estão descritos em "Definindo Filas Locais em um Provedor de Serviços" na página 46. Você deve especificar um valor adequado para STRINGS na definição de arquivo. O valor padrão 1 é improvável que seja suficiente e é recomendado que você use 10 em substituição.
3. Defina um recurso PROCESSTYPE com o nome DFHMQSOA, usando o nome do arquivo de repositório como o valor para a opção FILE.
4. Assegure que durante o processamento de uma mensagem persistente, um programa emita um comando **EXEC CICS SYNCPOINT** antes do primeiro ponto de sincronização implícito ser solicitado; por exemplo, usar um comando SPI tal como **EXEC CICS CREATE TDQUEUE** utiliza um ponto de sincronização implicitamente. A emissão de um comando **EXEC CICS SYNCPOINT** confirma que a mensagem persistente foi processada com sucesso. Se um programa não solicitar explicitamente um ponto de sincronização antes de tentar utilizar um ponto de sincronização implicitamente, um encerramento de forma anormal de ASP7 é emitido.

Resultados

O que Fazer Depois

Para mensagens de solicitação unidirecionais, se o serviço da web encerra de forma anormal ou é recuperado, informações suficientes são retidas para permitir que uma transação ou um programa tente novamente a solicitação com falha ou relate a falha apropriadamente. É necessário fornecer esta transação ou este programa de recuperação. Consulte “Processamento de Mensagem Persistente” para obter detalhes.

Processamento de Mensagem Persistente

Quando uma solicitação de serviço da web é recebida em uma mensagem persistente do WebSphere MQ, o CICS cria um processo BTS exclusivo com o tipo de processo DFHMQSOA. Os dados relacionados à solicitação de entrada são capturados em contêineres de dados BTS que estão associados ao processo.

O processo é, então, planejado para executar assincronicamente. Se o serviço da web é concluído com sucesso e confirmado, o CICS exclui o processo BTS. Isto inclui o caso em que uma falha de SOAP é gerada e retornada ao solicitante de serviço da web.

Processamento de Erro

Se ocorrer um erro ao criar o processo BTS necessário, a transação de serviço da web encerrará de forma anormal e a solicitação do serviço da web de entrada não será processada. Se o BTS não for utilizável, a mensagem DFHPI0117 será emitida e o CICS continuará sem BTS, usando o mecanismo do contêiner baseado em canal existente.

Se uma falha do CICS ocorrer antes do serviço da web iniciar ou concluir o processamento, a recuperação do BTS assegurará que o processo seja reagendado quando o CICS for reiniciado.

Se o serviço da web encerrar de forma anormal e voltar, o processo BTS será marcado como concluído com um status ABENDED. Para mensagens de solicitação que requerem uma resposta, uma falha de SOAP é retornada ao solicitante de serviço da web. O processo BTS é cancelado e o CICS não retém nenhuma informação sobre a solicitação com falha. O CICS emite a mensagem DFHBA0104 na fila de dados temporários CSBA e a mensagem DFHPI0117 na fila de dados temporários CPIO.

Para mensagens unidirecionais, não há uma maneira de retornar informações sobre a falha ao solicitante, portanto o processo BTS é retido em um estado COMPLETE ABENDED. O CICS emite a mensagem DFHBA0104 na fila de dados temporários CSBA e DFHPI0116 na fila de dados temporários CPIO.

É possível usar a transação de CBAM para exibir quaisquer processos COMPLETE ABENDED ou você pode fornecer uma transação de recuperação para verificar processos COMPLETE ABENDED do DFHMQSOA e executar a ação apropriada.

Por exemplo, sua transação de recuperação poderia:

1. Reconfigurar o processo BTS usando o comando **RESET ACQPROCESS**.
2. Emitir o comando **RUN ASYNC** para tentar novamente o serviço da web com falha. Isto poderá manter uma contagem de novas tentativas em um outro contêiner de dados no processo, para evitar falha repetida.

3. Use informações nos contêineres de dados associados para relatar o problema:
 - O contêiner de dados DFHMQORIGINALMSG contém a mensagem recebida do WebSphere MQ, que pode conter cabeçalhos RFH2.
 - O contêiner de dados DFHMQMSG contém a mensagem do WebSphere MQ com quaisquer cabeçalhos RFH2 removidos.
 - O contêiner de dados DFHMQDLQ contém o nome da fila de devoluções associada à mensagem original.
 - O contêiner de dados DFHMQCONT contém o bloco de controle MQMD do WebSphere MQ relacionado ao **MQ GET** para a mensagem original.

Interoperabilidade Entre o Assistente de Serviços da Web e o WSRR

O assistente de serviços da web do CICS pode interoperar com o IBM WebSphere Service Registry and Repository (WSRR). Use o WSRR para localizar serviços da web que você está solicitando mais rapidamente e impingir o controle de versão dos serviços da web que você está fornecendo.

Ambos DFHLS2WS e DFHWS2LS incluem parâmetros para interoperar com o WSRR. O DFHLS2WS também inclui um parâmetro opcional que permite incluir seus próprios metadados customizados no documento WSDL no WSRR.

Se desejar que o assistente de serviços da web se comunique seguramente com o WSRR, você poderá usar a criptografia Secure Socket Level (SSL). Ambos DFHLS2WS e DFHWS2LS incluem parâmetros para usar a criptografia SSL.

Para utilizar SSL com o assistente de serviços da web e o WSRR, consulte “Exemplo de Como Usar SSL com o Assistente de Serviços da Web e o WSRR”.

Exemplo de Como Usar SSL com o Assistente de Serviços da Web e o WSRR

É possível interoperar com segurança entre o assistente de serviços da web e um servidor IBM WebSphere Service Registry and Repository (WSRR) usando criptografia de Secure Socket Layer (SSL). Para usar criptografia de SSL, são necessários um armazenamento de chaves e um armazenamento de confiança; você também deve especificar determinados parâmetros no assistente de serviços da web.

Sobre Esta Tarefa

Conclua as etapas a seguir para usar a criptografia de SSL para interações entre o assistente de serviços da web e o WSRR.

Procedimento

1. Crie um armazenamento de chaves para suas chaves privadas e seus certificados de chave pública (PKC).
 - a. É possível criar um armazenamento de chaves usando um programa de configuração de chaves tal como o IBM Key Management Utility (iKeyman).
 - b. Especifique o parâmetro **SSL-KEYSTORE** em DFHWS2LS ou DFHLS2WS com o nome completo do armazenamento de chaves que você criou.
 - c. Opcional: Especifique o parâmetro **SSL-KEYPWD** em DFHWS2LS ou DFHLS2WS com a senha do armazenamento de chaves que você criou.

2. Crie um armazenamento de confiança para todos os seus certificados da autoridade de certificação (CA) de raiz confiável. Estes certificados são usados para estabelecer a confiança de quaisquer certificados de chave pública de entrada.
 - a. É possível criar um armazenamento de confiança usando um programa de configuração de chavez tal como o IBM Key Management Utility (iKeyman).
 - b. Especifique o parâmetro **SSL-TRUSTSTORE** em DFHWS2LS ou DFHLS2WS com o nome completo do armazenamento de confiança que você criou.
 - c. Opcional: Especifique o parâmetro **SSL-TRUSTPWD** em DFHWS2LS ou DFHLS2WS com a senha do armazenamento de confiança que você criou.
3. Teste se o assistente de serviços da web está apto a se comunicar com o WSRR usando a criptografia SSL.
 - a. É possível usar os arquivos de amostra fornecidos pelo IBM WebSphere Application Server para testar o assistente de serviços da web com o WSRR.
 - Os armazenamentos de chaves de amostra fornecidos pelo WebSphere Application Server são DummyClientKeyFile.jks e DummyServerKeyFile.jks.
 - Os armazenamentos de confiança de amostra fornecidos pelo WebSphere Application Server são DummyClientTrustFile.jks e DummyServerTrustFile.jks.
 - b. Substitua as chaves nos arquivos de chave de amostra e de armazenamento de confiança. Estas chaves são fornecidas com o WebSphere Application Server e devem ser substituídas por segurança.

Resultados

O assistente de serviços da web agora pode usar criptografia SSL para se comunicar com segurança com o WSRR em uma rede.

Criando a Infraestrutura de Serviços da Web

Para implementar um serviço da web no CICS, você deve criar a infraestrutura de transporte necessária e definir um ou mais pipelines que processarão suas solicitações de serviço da web. Geralmente, um pipeline pode processar solicitações para vários serviços da web diferentes e, quando você implementa um novo serviço da web em seu sistema CICS, é possível escolher usar um pipeline existente.

A Infraestrutura de Serviços da Web

Os aplicativos CICS em uma região do CICS podem fornecer um serviço para aplicativos que são externos a essa região, ou solicitar um serviço deles, usando um pipeline de serviços da web. Quando CICS é um provedor de serviços, o aplicativo CICS fornece um serviço ao aplicativo externo. Quando o CICS é um solicitante de serviço, o aplicativo externo fornece um serviço ao aplicativo CICS. Os pipelines de serviços da web podem ser configurados para usar o zEnterprise Application Assist Processor (zAAP) onde disponível.

CICS como um Provedor de Serviços

Para que o CICS forneça um serviço a um solicitante de serviço externo, ele deve receber a solicitação de serviço e transmiti-la por meio de um pipeline para o programa do aplicativo de destino. A resposta do aplicativo é retornada ao solicitante de serviço por meio do mesmo pipeline.

Figura 15 mostra uma configuração de exemplo da arquitetura e recursos que são necessários para processar uma solicitação a partir de um solicitante de serviço externo quando o CICS é um provedor de serviços que usa um pipeline Java.

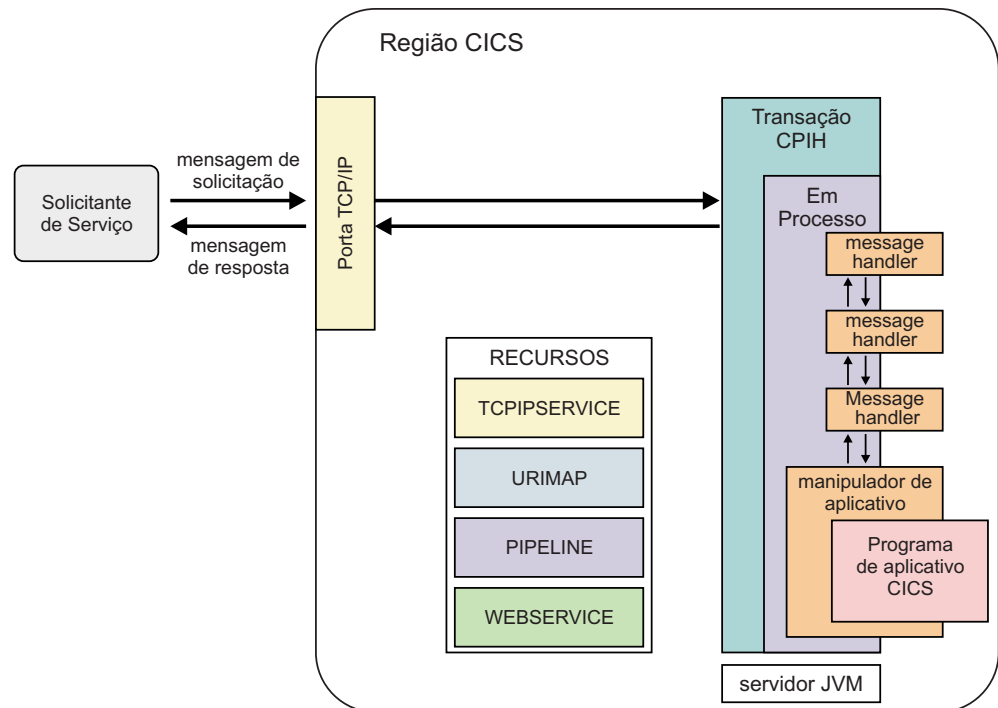


Figura 15. A Arquitetura e Recursos para um Provedor de Serviços

Para processar uma solicitação, o CICS deve executar as operações a seguir:

1. Receber a solicitação do solicitante de serviço.

O recurso TCPIPService especifica uma porta para solicitações recebidas. Esta porta é monitorada pela transação do listener de soquetes fornecida pelo CICS (CSOL).

2. Examinar a solicitação e extrair o conteúdo que é relevante para o programa aplicativo de destino.

Quando a mensagem de solicitação é recebida na porta apropriada, as definições de recurso URIMAP são varridas em busca de uma definição de URIMAP que possui seu atributo USAGE configurado como PIPELINE e seu atributo PATH configurado com o URI localizado na solicitação. Se uma definição de URIMAP apropriada é localizada, as definições de PIPELINE e WEBSERVICE dos atributos PIPELINE e WEBSERVICE da definição de URIMAP são usadas. O atributo TRANSACTION da definição de URIMAP determina o nome da transação que deve ser conectada para processar o pipeline. Por padrão, a transação CPIH é usada. A definição de URIMAP também identifica os recursos PIPELINE e WEBSERVICE para uso. Estes recursos controlam o processamento que o CICS executa.

3. Chamar o programa de aplicativo, transmitindo dados extraídos da solicitação.

Os manipuladores de mensagem no pipeline e no manipulador de aplicativo convertem a mensagem de solicitação na estrutura de linguagem do aplicativo que o programa de aplicativo do provedor de serviços espera. O programa processa esta entrada e retorna uma resposta ao manipulador de aplicativo.

4. Construir uma resposta usando dados retornados pelo programa de aplicativo e envie uma resposta ao solicitante de serviço.

O manipulador de aplicativo e os manipuladores de mensagem convertem a mensagem de resposta recebida do aplicativo do provedor de serviços em uma mensagem no formato da solicitação original. Esta mensagem é enviada de volta ao solicitante de serviço.

Alguns processamentos no pipeline podem ser executados usando o zEnterprise Application Assist Processor (zAAP) se o pipeline está configurado apropriadamente. Para obter mais informações, consulte “Pipelines SOAP Baseados em Java” na página 55.

CICS como um Solicitante de Serviço

Para CICS chamar um serviço externo, um programa de aplicativo envia uma solicitação que passa por um pipeline para um serviço de destino. A resposta do serviço é retornada para o programa de aplicativo por meio do mesmo pipeline.

Figura 16 mostra uma configuração de exemplo da arquitetura e recursos que são necessários para processar uma solicitação a partir de um programa de aplicativo CICS para dados a partir de um provedor de serviços que é externo à região do CICS, usando um pipeline Java.

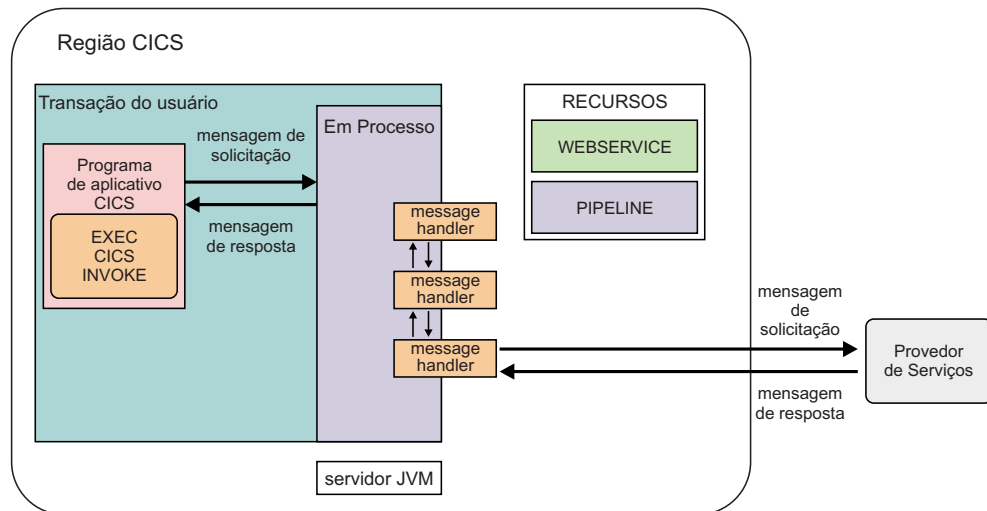


Figura 16. A Arquitetura e Recursos para um Solicitante de Serviço

Para processar uma solicitação, o CICS deve executar as operações a seguir:

1. Construir uma solicitação usando dados fornecidos pelo programa de aplicativo.

Quando o programa de aplicativo CICS inicia uma solicitação para um provedor de serviços que é externo à região do CICS, o aplicativo solicitante chama o comando EXEC CICS INVOKE SERVICE. O comando EXEC CICS INVOKE SERVICE chama o pipeline. O pipeline converte a estrutura de linguagem do aplicativo em uma linguagem que o provedor de serviços pode processar, por exemplo uma mensagem SOAP.

2. Enviar a solicitação ao provedor de serviços.

O CICS envia a mensagem de solicitação para o provedor de serviços remoto usando HTTP ou o WebSphere MQ.

3. Receber uma resposta do provedor de serviços.

Quando a mensagem de resposta do provedor de serviços é recebida, o CICS transmite a mensagem de volta ao pipeline.

4. Examinar a resposta e extrair o conteúdo que é relevante ao programa de aplicativo original.

O pipeline converte a mensagem de resposta do provedor de serviços na estrutura de linguagem do aplicativo, a qual é transmitida ao programa de aplicativo. O controle é, então, retornado ao programa de aplicativo.

Alguns processamentos no pipeline podem ser executados usando o zEnterprise Application Assist Processor (zAAP) se o pipeline está configurado apropriadamente. Para obter mais informações, consulte “Pipelines SOAP Baseados em Java”.

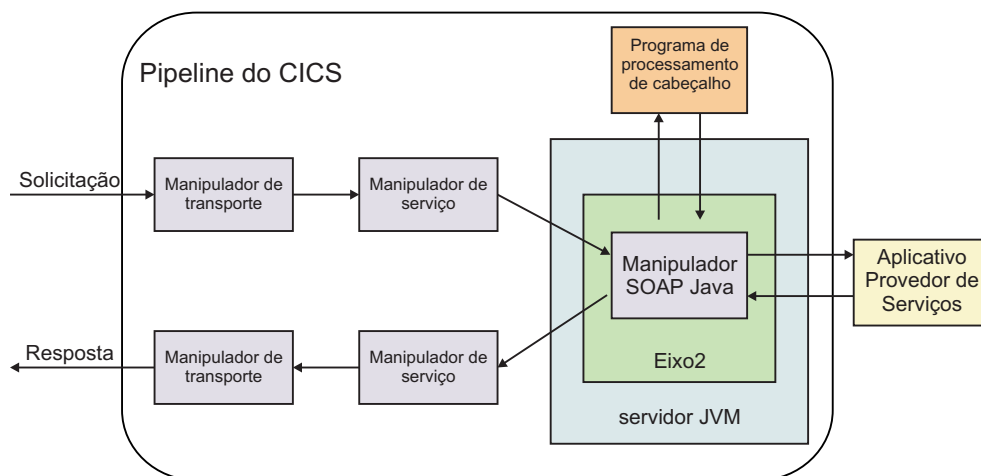
Pipelines SOAP Baseados em Java

O CICS suporta o uso do mecanismo SOAP baseado em Java Axis2 para processar solicitações de serviço da web em pipelines do provedor e do solicitante. Como o Axis2 usa Java, o processamento SOAP é elegível para transferência para o zEnterprise Application Assist Processor (zAAP).

Axis2 é um mecanismo do serviço da web de software livre da fundação Apache e é fornecido com o CICS para processar mensagens SOAP em um ambiente Java. É possível optar por usar Axis2 incluindo um manipulador SOAP Java em seu arquivo de configuração de pipeline e criando um servidor da JVM para manipular o processamento de Axis2.

A ativação de Axis2 não requer regeneração dos arquivos de ligação para qualquer serviço da web existente que usa o pipeline. Os tempos de resposta podem ser mais lentos ao usar o Axis2, mas é possível transferir o processamento de SOAP para zAAP. Para obter mais informações sobre a transferência para o zAAP, consulte Suporte Java em CICS.

Quando o CICS é um provedor de serviços, o manipulador de terminal baseado em Java usa Axis2 para analisar o envelope SOAP para uma mensagem de solicitação. É possível usar programas de programa de processamento de cabeçalho para processar quaisquer cabeçalhos SOAP associados à mensagem SOAP. Axis2 também constrói a mensagem de resposta SOAP. Este processo é mostrado no diagrama a seguir:



Quando o CICS é um solicitante de serviço, o manipulador inicial baseado em Java no pipeline usa Axis2 para gerar o envelope SOAP para uma mensagem de solicitação. É possível usar programas de programa de processamento de cabeçalho para processar quaisquer cabeçalhos SOAP associados à mensagem SOAP. Axis2 também analisa a mensagem de resposta SOAP.

Aplicativos de Serviço da Web e Java

Para pipelines SOAP no modo de provedor, mensagens de solicitação e resposta são transmitidas entre o manipulador de terminal do pipeline e o aplicativo de serviço da web usando um manipulador de aplicativo. O manipulador de aplicativo processa o corpo de uma solicitação SOAP para que a solicitação possa ser usada pelo aplicativo. O manipulador de aplicativo também gera uma resposta usando os retorno retornados do aplicativo. Se o manipulador de terminal de seu pipeline for um manipulador de mensagem baseado em Java, será possível especificar o manipulador de aplicativo Axis2 fornecido no arquivo de configuração de pipeline, em oposição à especificação do manipulador de aplicativo DFHPITP fornecido. O processamento do manipulador de aplicativo pode, então, ser transferido para o zAAP. Para obter mais informações sobre manipuladores de aplicativo, consulte “Manipuladores de Aplicativo” na página 90.

Para pipeline SOAP no modo do solicitante, o aplicativo de serviço da web chama o pipeline usando o comando **EXEC CICS INVOKE SERVICE**. As mensagens de solicitação e resposta são, então, transmitidas entre o aplicativo de serviço da web e o manipulador inicial no pipeline. Se especificar um manipulador baseado em Java como o manipulador inicial no pipeline, o comando **EXEC CICS INVOKE SERVICE** será processado por Axis2, tornando possível transferir este processo para zAAP. Se o primeiro manipulador não for um manipulador baseado em Java, o comando **EXEC CICS INVOKE SERVICE** será processado pelo CICS.

Processamento de Axis2 em um Servidor da JVM

Axis2 requer um servidor da JVM, o qual é representado por um recurso JVMSERVER no CICS. O servidor da JVM é um ambiente de tempo de execução que pode manipular diversas solicitações simultâneas de diferentes programas Java em uma única JVM. O caminho da classe para o servidor da JVM deve incluir os arquivos Java archive do Axis2. É possível incluir automaticamente todos os arquivos JAR necessários no caminho da classe especificando a opção `JAVA_PIPELINE` no perfil da JVM. O arquivo de configuração de pipeline também deve apontar para o recurso JVMSERVER que é configurado para suportar Axis2.

Para obter mais informações sobre servidores da JVM, consulte Suporte Java em CICS.

Manipuladores de Cabeçalho Axis2

Embora você possa usar programas de processamento de cabeçalho existentes, é mais eficiente gravar manipuladores Axis2 em Java para processar os cabeçalhos SOAP. Estes manipuladores também podem ser executados no servidor da JVM e são, portanto, elegíveis para transferência. Para obter mais informações sobre como criar manipuladores Axis2, consulte Gravando seu Próprio Módulo Axis2.

Um programa do manipulador de cabeçalho pode usar APIs Axis2 para modificar ou interagir com o ambiente do Axis2, mensagens SOAP e serviços da web individuais. Não use essas APIs para customizar o Axis2, pois você poderá mudar

o Axis2 de uma maneira que o CICS não possa executar o mecanismo corretamente. Manipuladores Axis2 são suportados somente se eles interagem com o ambiente do Axis2 de uma maneira compatível com como o CICS usa Axis2.

Repositório Axis2

O Axis2 usa um repositório para armazenar todos os seus arquivos de configuração, serviços e módulos. O CICS fornece um repositório padrão no diretório *usshome/lib/pipeline/repository* no z/OS UNIX, em que *usshome* é o valor do parâmetro de inicialização do sistema **USSHOME**.

O repositório padrão contém o arquivo de configuração, *axis2.xml*, que é requerido pelo CICS para usar o Axis2. Este arquivo está no subdiretório */conf* no repositório. Se você criar seu próprio repositório, deverá copiar este arquivo em seu repositório para que o CICS trabalhe com o Axis2.

Não edite o arquivo *axis2.xml*, a menos que esteja registrando programas do manipulador. Este arquivo é gerenciado como uma parte interna do CICS, portanto, você não deve fazer nenhuma outra mudança neste arquivo, a menos que solicitado para fazer isso pelo suporte IBM.

Formatação de dados para serviços da web

Diferentes tecnologias CICS podem gerar dados JSON e XML que são igualmente compatíveis com a especificação, mas fisicamente diferentes. Elas também podem relatar erros que são localizados em uma mensagem de entrada de diferentes maneiras, como resultado da ordem na qual elas aplicam verificações para validar os dados.

O CICS usa várias tecnologias diferentes para transformar automaticamente dados JSON e XML. Elas incluem z/OS Connect for CICS (variantes Java e não Java), Axis2 e recursos PIPELINE. Essas tecnologias geram dados JSON e XML em um formato externo, conforme determinado pelas especificações relevantes.

Pode haver várias maneiras de representar dados, que são igualmente compatíveis com a especificação, mas fisicamente diferentes. As tecnologias CICS sempre geram dados que são compatíveis, mas pode haver diferenças físicas entre eles. Por exemplo, se você alternar entre as opções Java e não Java do z/OS Connect for CICS para JSON, poderá detectar pequenas diferenças no JSON gerado.

Tais diferenças podem incluir mensagens de erro diferentes sendo relatadas sob condições de falha, diferenças no modo como caracteres de espaço em branco são inseridos, representações alternativas (porém equivalentes) para dados numéricos e variações no modo como os caracteres especiais são escapados. Outras mudanças dessa natureza podem ser introduzidas como resultado da aplicação de manutenção corretiva no CICS ou com novas liberações do CICS.

Sistemas parceiro, tal como um cliente JSON, devem ser gravados para tolerar as variações compatíveis com a especificação desta natureza. Parceiros geralmente exploram bibliotecas de vários segmentos de mercados amplamente usadas e tecnologias para interagir com JSON e XML; essas bibliotecas manipulam automaticamente essas pequenas diferenças de formatação. No entanto, é possível para sistemas parceiros menos compatíveis detectar e responder de forma diferente às diferenças de formatação nas várias tecnologias CICS, portanto, é necessário cuidado se você está gravando aplicativos que trabalham diretamente com o JSON ou XML sem o benefício de um analisador baseado em padrões.

CICS como um provedor de serviços para solicitações JSON

Para o CICS fornecer um serviço para um cliente JSON externo, ele deve receber a solicitação e transmiti-la por meio de um pipeline para o programa de aplicativo de destino. A resposta do aplicativo é retornada ao cliente JSON por meio do mesmo pipeline.

Há várias maneiras de configurar o CICS como um provedor de serviços para solicitações JSON:

- Usando o z/OS Connect for CICS.

O z/OS Connect for CICS é uma tecnologia para acessar ativos z/OS, tais como programas CICS, usando JSON. Para obter mais informações sobre o z/OS Connect for z/OS e algumas das diferenças entre o z/OS Connect for CICS e os pipelines Java do CICS, consulte Introdução ao z/OS Connect.

- Usando os pipelines Java do CICS.

Os pipelines Java do CICS são a tecnologia fornecida em versões anteriores do CICS para permitir acesso aos programas CICS usando JSON. Para obter mais informações sobre os pipelines Java do CICS, consulte Pipelines SOAP Baseados em Java.

- Usando os recursos JAX-RS e JSON do servidor JVM CICS Liberty diretamente.

- Usando o manipulador de terminal fornecido pelo CICS, DFHPIJT.

É possível configurar um pipeline do provedor com o manipulador de terminal DFHPIJT. Isso permite que o pipeline processe solicitações JSON sem a necessidade de instalar um servidor JVM. As restrições a seguir se aplicam:

Restrição:

- Os serviços da web JSON RESTful não são suportados.
- O comutador de contexto no pipeline não é suportado.
- Não é possível usar serviços da web SOAP e JSON em um pipeline JSON. DFHPIJT manipula somente mensagens JSON. O recebimento de uma mensagem SOAP resulta em uma resposta de erro.

O CICS recebe dados JSON e os transforma em dados do aplicativo estruturados que são entendidos pelo programa de aplicativo CICS. As respostas do aplicativo CICS são transformadas em uma carga útil JSON para a resposta de saída. As transformações requerem a análise das mensagens. Se você usar o z/OS Connect for CICS, terá a opção de usar um analisador JSON baseado em Java ou um equivalente não Java. A opção de configuração é especificada no arquivo de configuração de pipeline do z/OS Connect for CICS. Para obter informações adicionais sobre como configurar o z/OS Connect for CICS, consulte Configurando o z/OS Connect for CICS. Se você usar os pipelines Java do CICS, a análise será sempre executada somente usando Java no servidor JVM. As implicações das diferentes decisões de configuração são:

- Se você analisar o JSON usando Java, o processamento será elegível para transferência para um zEnterprise Application Assist Processor (zAAP) se ele estiver disponível. A transferência do processamento pode ter custo-benefício.
- Se você usar o analisador não Java do z/OS Connect for CICS, algumas cargas de trabalho poderão ter benefícios de desempenho e rendimento. Para obter mais informações sobre quais cargas de trabalho podem se beneficiar, consulte o material de desempenho que é disponibilizado com esta liberação. Mesmo se você utilizar o analisador não Java, a maioria do processamento de infraestrutura do z/OS Connect for CICS será elegível para transferência para zAAP.

- Se você usar o manipulador de terminal fornecido pelo CICS, DFHPIJT, algumas cargas de trabalho poderão ter benefícios de desempenho e rendimento. Quando você usa esse método de processamento de solicitações JSON, nenhum processamento é elegível para transferência para zAAP.

O CICS como um provedor de serviços para solicitações JSON usando z/OS Connect for CICS:

Para o CICS fornecer um serviço a um cliente JSON externo, ele deve receber a solicitação em um z/OS Connect for CICS, transformar a mensagem JSON e transmiti-la ao programa de aplicativo de destino. A resposta do aplicativo é retornada ao cliente JSON por meio do mesmo mecanismo.

O Figura 17 mostra uma configuração de exemplo da arquitetura e recursos que são necessários para processar uma solicitação a partir de um cliente JSON externo quando o CICS é um provedor de serviços que usa z/OS Connect para CICS.

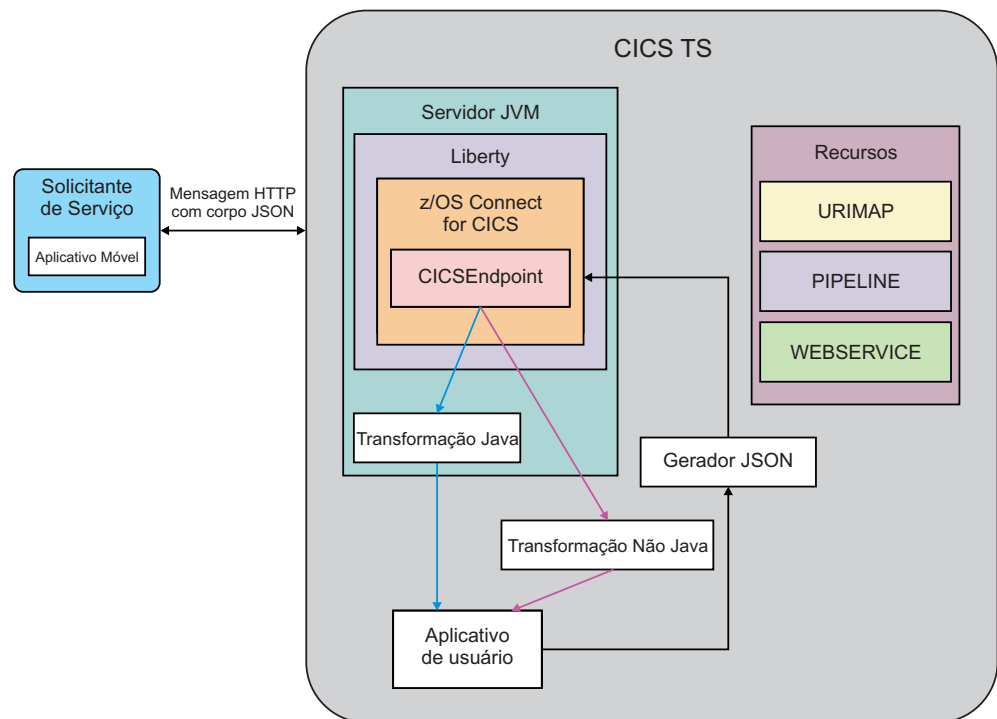


Figura 17. A arquitetura e os recursos para um provedor de serviços JSON que usa o z/OS Connect for CICS

Processando uma solicitação JSON com o z/OS Connect for CICS

Para processar uma solicitação, o CICS conclui as operações a seguir:

1. Receber a solicitação do solicitante de serviço.
O WebSphere Liberty recebe a solicitação e a transmite para o z/OS Connect for CICS.
2. O CICS resolve o recurso URIMAP para a solicitação, varrendo as definições de URIMAP com um atributo USAGE configurado no JVMSERVER. O URIMAP

identifica o recurso WEBSERVICE que é usado. Uma nova tarefa do CICS é iniciada para processar a solicitação usando o ID de transação do URIMAP. Por padrão, a transação CPIH é usada.

O recurso WEBSERVICE controla o processamento que o CICS executa. Em particular, o arquivo WSBInd apontado pelo recurso WEBSERVICE é usado para transformação de dados entre JSON e dados de aplicativos estruturados. Arquivos WSBInd para serviços da web JSON são gerados usando os utilitários DFHLS2JS e DFHJS2LS.

Nota: A validação de tempo de execução de dados JSON no esquema não é suportada. O valor do atributo VALIDATION de um recurso WEBSERVICE que é usado com uma carga útil JSON é ignorado.

Para obter informações sobre quaisquer restrições que se aplicam, consulte Restrições de serviço da web JSON.

3. O z/OS Connect for CICS processa a solicitação de acordo com a maneira como ele é configurado. Se o z/OS Connect for CICS é configurado para usar interceptores globais, os interceptores são executados durante esse processamento.
4. O CICSEndpoint recebe o controle. A carga útil JSON é transformada em dados de aplicativos estruturados de acordo com a opção de configuração especificada no arquivo de configuração de pipeline. Há duas opções para como a transformação é executada:
 - A transformação Java é executada no servidor JVM. Este processamento é elegível para transferência para um processador zAAP se um está disponível.
 - A transformação não Java é executada fora do servidor JVM e pode fornecer benefícios de desempenho e rendimento para determinadas cargas de trabalho. Para obter mais informações sobre quais cargas de trabalho podem se beneficiar, consulte o material de desempenho que é disponibilizado com esta liberação.

O mapeamento é executado de acordo com as informações no arquivo WSBInd. A saída da transformação é equivalente em ambos os casos.

5. O CICS se vincula ao programa de aplicativo e transmite os dados transformados. O programa processa esta entrada e retorna uma resposta ao gerador JSON.
6. O gerador JSON gera uma mensagem de resposta JSON usando a saída da etapa 5. Esta mensagem é enviada de volta ao solicitante de serviço por meio do z/OS Connect for CICS.

O z/OS Connect for CICS também fornece uma interface RESTful que suporta os métodos RESTful padrão:

POST
PUT
GET
DELETE
HEAD

Quando apropriado, esta interface usa a transformação e o gerador para o qual o pipeline está configurado, da mesma maneira que faz solicitações de JSON normais. Para obter mais informações sobre serviços da web JSON RESTful, consulte Conceitos de serviços da web JSON RESTful.

A maioria do processamento de infraestrutura do z/OS Connect for CICS é elegível para transferência para um zEnterprise Application Assist Processor (zAAP).

CICS como um provedor de serviços para solicitações JSON usando pipelines Java do CICS:

Para o CICS fornecer um serviço para um cliente JSON externo, ele deve receber a solicitação e transmiti-la por meio de um pipeline para o programa de aplicativo de destino. A resposta do aplicativo é retornada ao cliente JSON por meio do mesmo pipeline. A conversão de JSON é executada usando Java no servidor JVM.

Figura 18 mostra uma configuração de exemplo da arquitetura e dos recursos que são necessários para processar uma solicitação a partir de um cliente JSON externo quando o CICS é um provedor de serviços que usa um pipeline Java. O processamento de pipeline para uma solicitação JSON é semelhante à maneira que o CICS processa uma solicitação SOAP em um pipeline Java. Para obter mais informações, consulte Pipelines SOAP Baseados em Java.

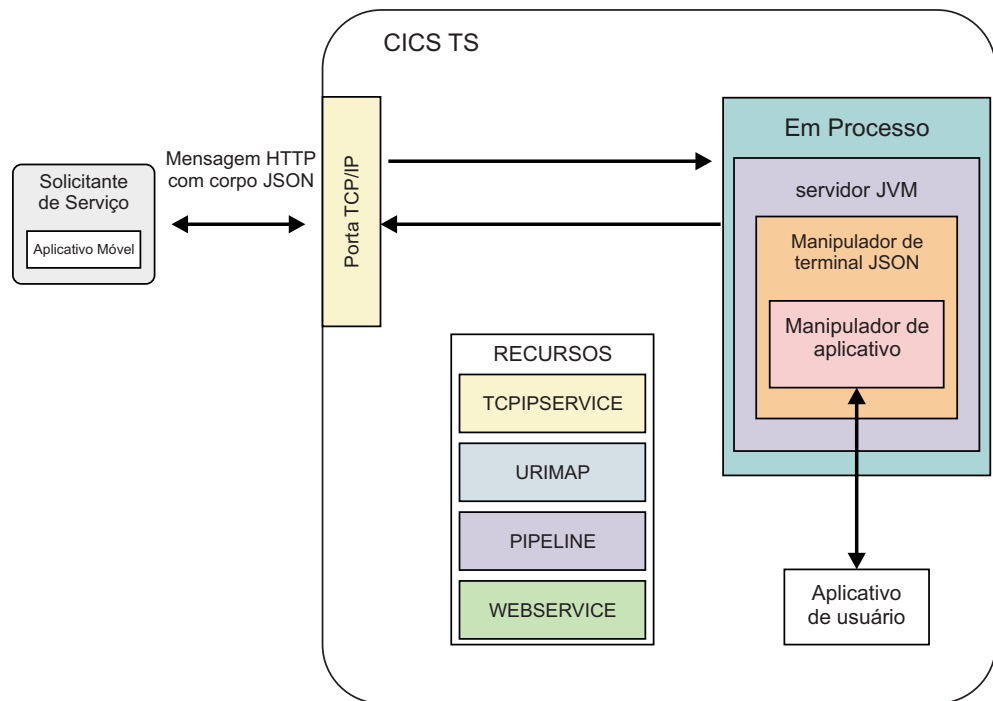


Figura 18. A arquitetura e os recursos para um provedor de serviços JSON que usa um pipeline Java do CICS

Processando uma solicitação JSON

Para processar uma solicitação, o CICS conclui as operações a seguir:

1. Receber a solicitação do solicitante de serviço.

O recurso TCPIP SERVICE especifica uma porta para solicitações recebidas. Esta porta é monitorada pela tarefa do listener de soquetes fornecida pelo CICS (CSOL).

2. Examinar a solicitação e extrair o conteúdo que é relevante para o programa aplicativo de destino.

Quando a mensagem de solicitação é recebida na porta apropriada, as definições de recurso URIMAP são varridas em busca de uma definição de URIMAP que possui seu atributo USAGE configurado como PIPELINE e seu atributo PATH configurado com o URI localizado na solicitação. Se uma definição de URIMAP apropriada é localizada, as definições de PIPELINE e WEBSERVICE dos atributos PIPELINE e WEBSERVICE da definição de URIMAP são usadas. O atributo TRANSACTION da definição de URIMAP determina o nome da transação que deve ser conectada para processar o pipeline. Por padrão, a transação CPH é usada. A definição de URIMAP também identifica os recursos PIPELINE e WEBSERVICE para uso.

Esses recursos PIPELINE e WEBSERVICE controlam o processamento executado pelo CICS. Em particular, o arquivo WSBIND apontado pelo recurso WEBSERVICE é usado para transformação de dados entre JSON e estruturas de linguagem. Arquivos WSBIND para serviços da web JSON são gerados usando os utilitários DFHLS2JS e DFHJS2LS.

Nota: A validação de tempo de execução de dados JSON no esquema não é suportada. O valor do atributo VALIDATION de um recurso WEBSERVICE que é usado com uma carga útil JSON é ignorado.

Para obter informações sobre quaisquer restrições que se aplicam, consulte Restrições de serviço da web JSON.

3. O processamento de pipeline começa e a solicitação flui por meio de quaisquer manipuladores que estejam definidos. Não é esperado que qualquer um dos manipuladores que são atualmente fornecidos pelo CICS para serviços da web SOAP sejam relevantes para serviços da web JSON.
4. No final do pipeline, o manipulador de terminal JSON é chamado. Este manipulador de terminal é um programa Java que cria interface com o pipeline Axis2. O manipulador de terminal executa a definição necessária da configuração Axis2 e, em seguida, inicia o mecanismo Axis2 com o corpo da solicitação de HTTP. Dentro do pipeline Axis2, o corpo JSON (se presente) é analisado e um modelo de objeto Java que representa o conteúdo é construído. O CICS, então, chama o manipulador de aplicativo. A principal função do manipulador de aplicativo é mapear a representação do modelo de objeto Java da solicitação para dados do aplicativo. Esse mapeamento é executado usando a descrição da estrutura de linguagem no arquivo WSBIND.
5. Chame o programa de aplicativo transmitindo dados que são extraídos da solicitação.
Em seguida, o manipulador de aplicativo vincula-se ao programa de aplicativo. O programa processa esta entrada e retorna uma resposta ao manipulador de aplicativo.
6. Construa uma resposta usando dados retornados pelo programa de aplicativo e envie uma resposta ao solicitante de serviço.
O manipulador de aplicativo e os manipuladores de mensagem convertem a mensagem de resposta recebida do aplicativo do provedor de serviços em uma mensagem no formato da solicitação original. Esta mensagem é enviada de volta ao solicitante de serviço.

Alguns dos processamentos no pipeline são elegíveis para transferência para um zEnterprise Application Assist Processor (zAAP).

Criando a infraestrutura do CICS para um provedor de serviços SOAP

Para criar a infraestrutura do CICS para um provedor de serviços SOAP, deve-se criar um arquivo de configuração de pipeline e criar inúmeros recursos do CICS.

Antes de Iniciar

Se desejar usar um pipeline Java, assegure que um recurso do JVMSERVER exista com a opção JAVA_PIPELINE=YES especificada no Perfil da JVM.

Um servidor da JVM pode manipular o processamento de SOAP para muitos pipelines Java.

Sobre Esta Tarefa

É possível definir o recurso PIPELINE em uma região CICS local usando as funções do CICS ou CICSplex SM ou você pode usar o CICS Explorer para definir o recurso PIPELINE em uma região CICS local ou em um pacote configurável do CICS. Ao usar o CICS Explorer para definir um recurso PIPELINE em um pacote configurável do CICS, você também cria o arquivo de configuração de pipeline e o empacota no pacote configurável do CICS, de modo que não seja preciso gerenciar este arquivo separadamente. Os recursos PROGRAM e WEBSERVICE também podem ser definidos nos pacotes configuráveis do CICS. Ao definir um recurso WEBSERVICE no pacote configurável do CICS, é possível importar um arquivo de ligação de serviço da web e um documento WSDL ou archive WSDL e inclui-los no pacote configurável. Também é possível criar definições de URIMAP para suportar o serviço da web e empacotá-los em um pacote configurável. Para obter mais ajuda com o uso do CICS Explorer para criar e editar recursos em pacotes configuráveis CICS, consulte Trabalhando com pacotes configuráveis na documentação do produto CICS Explorer.

Procedimento

1. Defina a infraestrutura de transportar.
 - a. Se estiver usando o transporte do WebSphere MQ, você deverá definir uma ou mais filas locais que armazenam mensagens de entrada até elas serem processadas e um processo do acionador que especifica a transação do CICS que processará as mensagens de entrada.
 - 1) Consulte Configurando o CICS para utilizar o transporte do WebSphere MQ para obter detalhes.
 - b. Se estiver usando o transporte HTTP, você deverá definir um recurso TCPIPSERVICE que define a porta na qual solicitações de entrada serão recebidas.
 - 1) Consulte Recursos CICS para Serviços da Web para obter detalhes.
2. Opcional: Repita esta etapa para cada configuração de transporte diferente necessária.
3. Defina os manipuladores de mensagem e os programas de processamento de cabeçalho que deseja incluir no arquivo de configuração de pipeline para processar solicitações de serviço da web de entrada e suas respostas. O CICS fornece os manipuladores e programas de processamento de cabeçalho a seguir:
 - a. Manipuladores de mensagem SOAP, para processar mensagens SOAP 1.1 ou 1.2. É possível suportar somente um nível de SOAP em um pipeline do provedor de serviços.
 - b. Manipulador de MTOM, para processar mensagens MIME Multiparte/Relacionadas que estão em conformidade com as especificações MTOM/XOP.
 - c. Suporte para proteger serviços da web, para processar mensagens de serviço da web seguras.

- d. Suporte para transações de serviços da web, para processar mensagens da transação atômica.
4. Opcional: Se desejar executar seu próprio processamento no pipeline, você deverá criar um manipulador de mensagem ou programa de processamento de cabeçalho. Consulte Manipuladores de mensagens para obter detalhes. Se você decidir criar programas de manipulador de mensagem customizados, para otimizar o desempenho você deve torná-los thread-safe.
5. Crie um arquivo de configuração de pipeline XML contendo seus manipuladores de mensagem, programas de processamento de cabeçalho e o manipulador de aplicativo.
 - a. O CICS fornece duas amostras de arquivo de configuração de pipeline no modo de provedor básicas, `basicsoap11provider.xml` e `basicsoap11javaprovider.xml`.
 - b. É possível editar estas amostras ou incluir manipuladores de mensagem adicionais conforme apropriado. As amostras são fornecidas na biblioteca `/usr/lpp/cicsts/cicsts54/samples/pipelines` (em que `/usr/lpp/cicsts/cicsts54` é o diretório de instalação padrão para arquivos CICS no z/OS UNIX).
 - c. Para obter mais informações sobre as opções disponíveis no arquivo de configuração de pipeline, consulte Arquivos de configuração de pipeline
6. Copie o arquivo de configuração de pipeline para um diretório adequado no z/OS UNIX.
7. Altere as permissões do arquivo de configuração de pipeline para permitir que a região do CICS leia o arquivo.
8. Repita as etapas 5 a 7 para cada configuração de pipeline diferente requerida.
9. Crie um recurso PIPELINE.
 - a. O recurso PIPELINE define o local do arquivo de configuração de pipeline. Ele também especifica um *diretório de recebimento*, que é o diretório do z/OS UNIX que contém os arquivos de ligação de serviço da web e, opcionalmente, o WSDL.
 - b. Repita esta etapa para cada configuração de pipeline diferente.
- a. Ao criar um recurso de PIPELINE, o CICS lê quaisquer arquivos no diretório de recebimento especificado e cria o recurso WEBSERVICE e o recurso URIMAP dinamicamente.
10. A menos que você use definições de PROGRAM instaladas automaticamente, crie um recurso PROGRAM para cada programa que é executado no pipeline. Elas incluem o programa aplicativo de destino, que normalmente é executado sob a transação CPIH. A transação é definida com o atributo TASKDATALOC(ANY). Portanto, quando você edita o link do programa, deve especificar a opção AMODE(31).

Resultados

Seu sistema CICS agora contém a infraestrutura necessária para cada provedor de serviços.

O que Fazer Depois

É possível estender a configuração quando precisar fazer isso, seja para definir a infraestrutura de transporte adicional ou para criar pipelines adicionais.

Criando a infraestrutura do CICS para um solicitante de serviço SOAP

Para criar a infraestrutura do CICS para um solicitante de serviço SOAP, deve-se criar um arquivo de configuração de pipeline e criar um número de recursos do CICS.

Antes de Iniciar

Se desejar usar um pipeline Java, assegure-se de que exista um recurso JVMSERVER com a opção `JAVA_PIPELINE=YES` especificada no Perfil da JVM. Consulte Recursos JVMSERVER.

Um servidor de JVM pode manipular o processamento de SOAP para muitos pipelines Java.

Sobre Esta Tarefa

É possível definir o recurso PIPELINE em uma região CICS local usando funções do CICS ou do CICSplex SM ou você pode usar o CICS Explorer para definir o recurso PIPELINE em uma região CICS local ou em um pacote configurável do CICS. Ao usar o CICS Explorer para definir um recurso PIPELINE em um pacote configurável do CICS, você também cria o arquivo de configuração de pipeline e o empacota no pacote configurável do CICS, de modo que não seja preciso gerenciar este arquivo separadamente. Os recursos PROGRAM, WEBSERVICE e URIMAP também podem ser definidos nos pacotes configuráveis do CICS. Ao definir um recurso WEBSERVICE em um pacote configurável do CICS, é possível importar um arquivo de ligação de serviço da web e um documento WSDL ou archive WSDL e empacotá-los no pacote configurável e, para um provedor de serviços, você pode optar por incluir uma definição de PROGRAM. Também é possível criar definições de URIMAP para suportar o serviço da web e empacotá-los em um pacote configurável. Para obter mais ajuda com o uso do CICS Explorer para criar e editar recursos em pacotes configuráveis CICS, consulte Trabalhando com pacotes configuráveis na documentação do produto CICS Explorer.

Procedimento

1. Defina os manipuladores de mensagem e os programas de processamento de cabeçalho que deseja incluir no arquivo de configuração de pipeline para processar solicitações de serviço da web de entrada e suas respostas. O CICS fornece os seguintes manipuladores e programas de processamento de cabeçalho:
 - a. Manipuladores de mensagem SOAP, para processar mensagens SOAP 1.1 ou 1.2. É possível suportar somente um nível de SOAP em um pipeline do solicitante de serviço.
 - b. Manipulador MTOM, para processar mensagens MIME Multiparte/Relacionadas que estão em conformidade com as especificações MTOM/XOP.
 - c. Manipulador de segurança, para processar mensagens de serviço da web seguras.
 - d. Programa de processamento de cabeçalho WS-AT, para processar mensagens da transação atômica.
2. Opcional: Se desejar executar seu próprio processamento no pipeline, você deverá criar um manipulador de mensagem ou programa de processamento de cabeçalho. Consulte “Manipuladores de mensagens” na página 134 para obter

detalhes. Se você decidir criar programas de manipulador de mensagem customizados, para otimizar o desempenho você deve torná-los thread-safe.

3. Crie um arquivo de configuração de pipeline XML contendo os manipuladores de mensagens e programas de processamento de cabeçalho. O CICS fornece duas amostras do arquivo de configuração de pipeline no modo do solicitante básico, `basicsoap11provider.xml` e `basicsoap11javaprovider.xml`, que podem ser copiadas e editadas conforme apropriado. Estas amostras são fornecidas na biblioteca `/usr/lpp/cicsts/cicsts54/samples/pipelines` (em que `/usr/lpp/cicsts/cicsts54` é o diretório de instalação padrão para arquivos CICS no z/OS UNIX). Para obter mais informações sobre as opções disponíveis no arquivo de configuração de pipeline, consulte “Arquivos de configuração de pipeline” na página 79
4. Copie o arquivo de configuração de pipeline para um diretório adequado no z/OS UNIX.
5. Altere as permissões do arquivo de configuração de pipeline para permitir que a região do CICS leia o arquivo.
6. Repita as etapas 3 a 5 para cada configuração de pipeline diferente requerida.
7. Crie um recurso PIPELINE. Consulte Recursos PIPELINE. O recurso PIPELINE define o local do arquivo de configuração de pipeline. Ele também especifica um *diretório de recebimento*, que é o diretório do z/OS UNIX que contém os arquivos de ligação de serviço da web e, opcionalmente, o WSDL. Também é possível especificar um tempo limite em segundos, que determina por quanto tempo o CICS aguarda por uma resposta a partir de provedores de serviço da web. Repita esta etapa para cada arquivo de configuração de pipeline. Ao criar um recurso PIPELINE, o CICS lê todos os arquivos no diretório de recebimento especificado e cria os recursos WEBSERVICE dinamicamente (consulte Recursos WEBSERVICE).
8. A menos que você use as definições PROGRAM de instalação automática, crie um recurso PROGRAM para cada programa que é executado no pipeline. Consulte Recursos PROGRAM. Estes programas incluem o programa aplicativo do solicitante de serviços, que normalmente é executado sob a transação CPIH. A transação é definida com o atributo `TASKDATALOC(ANY)`. Portanto, quando editar o link do programa, você deverá especificar a opção `AMODE(31)`.
9. Opcional: Crie um recurso URIMAP (consulte Recursos URIMAP) para solicitações do cliente para cada URI que os solicitantes de serviços usam para fazer solicitações, seguindo as instruções em Criando um recurso URIMAP para CICS como um cliente HTTP. É possível especificar o URI diretamente no comando **INVOKE SERVICE** em seus programas, em vez de usar um recurso URIMAP. Entretanto, usar um recurso URIMAP significa que você não precisa recompilar seus aplicativos se o URI de um provedor de serviços for alterado. Com um recurso URIMAP também é possível escolher implementar a definição do conjunto de conexões, na qual o CICS mantém a conexão do cliente aberta após o uso, para que ela possa ser reutilizada pelo aplicativo para solicitações subsequentes ou por um outro aplicativo que chama o mesmo serviço.

Resultados

Seu sistema CICS agora contém a infraestrutura necessária para cada solicitante de serviço.

O que Fazer Depois

É possível estender a configuração quando precisar fazer isso, para criar pipelines adicionais.

Criando a infraestrutura do CICS para um provedor de serviços JSON

Para criar a infraestrutura do CICS para um provedor de serviços JSON, você deverá criar um arquivo de configuração de pipeline e criar um número de recursos do CICS.

Antes de Iniciar

Para usar o CICS como um provedor de serviços para solicitações JSON ou usar a interface vinculável para transformar o JSON, defina e instale um recurso JVMSERVER com um perfil de JVM que tenha a opção **JAVA_PIPELINE=YES** especificada. Uma definição de recurso JVMSERVER de exemplo chamada DFHAXIS é fornecida no grupo DFH\$AXIS.

Nota: A infraestrutura descrita aqui assume que você não está usando o z/OS Connect for CICS para se conectar ao seu provedor de serviços JSON e, conseqüentemente, usa a análise sintática de Java no servidor JVM para analisar as mensagens JSON. Se você deseja usar análise JSON não Java, deve usar o z/OS Connect for CICS para se conectar ao serviço da web JSON. Para obter mais informações sobre como configurar o z/OS Connect for CICS, consulte *Configurando o z/OS Connect for CICS*.

Procedimento

1. Defina a infraestrutura de transportar.
Defina um recurso TCPIP SERVICE que define a porta na qual solicitações de entrada são recebidas. Consulte Recursos do CICS para serviços da web para obter detalhes.
2. Defina os manipuladores de mensagem que você deseja incluir no arquivo de configuração de pipeline para processar solicitações de serviço da web de entrada e suas respostas.
Se desejar executar seu próprio processamento no pipeline, você deverá criar um manipulador de mensagem. Consulte Manipuladores de mensagem para obter detalhes. Se você decidir criar programas de manipulador de mensagem customizados, para otimizar o desempenho você deve torná-los thread-safe.
3. Crie um arquivo de configuração de pipeline XML contendo seus manipuladores de mensagem, programas de processamento de cabeçalho e o manipulador de aplicativo. O CICS fornece uma amostra do arquivo de configuração de pipeline de modo de provedor básico, `jsonjavaprovider.xml`. É possível editar essa amostra para incluir manipuladores de mensagem adicionais conforme apropriado. Esta amostra é fornecida no diretório `/usr/lpp/cicsts/cicsts54/samples/pipelines`, em que `/usr/lpp/cicsts/cicsts54` é o diretório de instalação padrão para arquivos CICS no z/OS UNIX). Para obter mais informações sobre as opções disponíveis no arquivo de configuração de pipeline, consulte “Elementos Usados em Pipelines do Provedor de Serviços e do Solicitante de Serviço” na página 96.
4. Copie o arquivo de configuração de pipeline para um diretório adequado no z/OS UNIX.
5. Altere as permissões do arquivo de configuração de pipeline para permitir que a região do CICS leia o arquivo.
6. Crie um recurso PIPELINE. O recurso PIPELINE define o local do arquivo de configuração de pipeline. Ele também especifica um *diretório de recebimento*, que é o diretório z/OS UNIX que contém os arquivos de ligação de serviço da web. Repita esta etapa para cada configuração de pipeline diferente. Ao instalar um

recurso PIPELINE ou executar um PIPELINE SCAN, o CICS lê os arquivos .wsbind no diretório de recebimento especificado e cria os recursos WEBSERVICE e URIMAP apropriados dinamicamente.

7. A menos que você use definições de PROGRAM instaladas automaticamente, crie um recurso PROGRAM para cada programa que é executado no pipeline. Elas incluem o programa aplicativo de destino, que normalmente é executado sob a transação CPIH. A transação é definida com o atributo TASKDATALOC (ANY). Portanto, quando você edita o link do programa, deve especificar a opção AMODE(31).

Resultados

Você criou a infraestrutura necessária para cada provedor de serviços e agora pode instalar esses recursos em seu sistema CICS.

O que Fazer Depois

Instale os recursos. É possível estender a configuração quando precisar fazer isso, seja para definir a infraestrutura de transporte adicional ou para criar pipelines adicionais.

Criando a infraestrutura do CICS para um provedor de serviços JSON não Java

É possível configurar um ambiente não Java para processamento de solicitações de JSON. Para criar a infraestrutura do CICS para um provedor de serviços JSON não Java, deve-se criar um arquivo de configuração de pipeline e criar vários recursos do CICS.

Procedimento

1. Defina a infraestrutura de transportar.
Defina um recurso TCPIPService que define a porta na qual solicitações de entrada são recebidas. Consulte Recursos do CICS para serviços da web para obter detalhes.
2. Defina os manipuladores de mensagem que você deseja incluir no arquivo de configuração de pipeline para processar solicitações de serviço da web de entrada e suas respostas.
Se desejar executar seu próprio processamento no pipeline, você deverá criar um manipulador de mensagem. Consulte Manipuladores de mensagem para obter detalhes. Se você decidir criar programas de manipulador de mensagem customizados, para otimizar o desempenho você deve torná-los thread-safe.
3. Crie um arquivo de configuração de pipeline XML que contenha seus manipuladores de mensagem.
No arquivo de configuração, deve-se especificar o programa manipulador de terminal DFHPIJT em um elemento <terminal_handler>, conforme mostrado em Figura 19 na página 69. DFHPIJT é o programa do manipulador JSON fornecido pelo CICS que permite o processamento não Java de mensagens JSON.

```

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

```

Figura 19. Especificando o manipulador de terminal DFHPIJT para processamento não Java de mensagens JSON

Nota: Quando você usar DFHPIJT como o manipulador de terminal, não defina um manipulador de aplicativo no arquivo de configuração de pipeline, ou seja, o arquivo de configuração de pipeline não deve conter um elemento <apphandler>. Se um manipulador de aplicativo for especificado, ele não será chamado.

Para obter mais informações sobre as opções disponíveis no arquivo de configuração de pipeline, consulte “Elementos Usados em Pipelines do Provedor de Serviços e do Solicitante de Serviço” na página 96.

4. Copie o arquivo de configuração de pipeline para um diretório adequado no z/OS UNIX.
5. Altere as permissões do arquivo de configuração de pipeline para permitir que a região do CICS leia o arquivo.
6. Crie um recurso PIPELINE. O recurso PIPELINE define o local do arquivo de configuração de pipeline. Ele também especifica um *diretório de recebimento*, que é o diretório z/OS UNIX que contém os arquivos de ligação de serviço da web. Repita esta etapa para cada configuração de pipeline diferente. Ao instalar um recurso PIPELINE ou executar um PIPELINE SCAN, o CICS lê os arquivos .wsbind no diretório de recebimento especificado e cria os recursos WEBSERVICE e URIMAP apropriados dinamicamente.
7. A menos que você use definições de PROGRAM instaladas automaticamente, crie um recurso PROGRAM para cada programa que é executado no pipeline. Elas incluem o programa aplicativo de destino, que normalmente é executado sob a transação CPIH. A transação é definida com o atributo TASKDATALOC(ANY). Portanto, quando você edita o link do programa, deve especificar a opção AMODE(31).

Resultados

Você criou a infraestrutura necessária para cada provedor de serviços e agora pode instalar esses recursos em seu sistema CICS.

O que Fazer Depois

Instale os recursos. É possível estender a configuração quando precisar fazer isso, seja para definir a infraestrutura de transporte adicional ou para criar pipelines adicionais.

Configurando o z/OS Connect for CICS

Configurando o z/OS Connect for CICS 1.0

O z/OS Connect for CICS 1.0 é distribuído como parte do CICS Transaction Server. Deve-se configurar um servidor JVM e definir a configuração de pipeline e os recursos para z/OS Connect antes de poder implementar serviços JSON. Esta configuração inicial é uma atividade única.

Antes de Iniciar

Você já possui um servidor JVM do WebSphere Liberty que está configurado no CICS. Embora seja possível hospedar o z/OS Connect e outros serviços não relacionados no mesmo ambiente do WebSphere Liberty, é uma boa prática configurar um servidor JVM separado para o uso exclusivo do z/OS Connect. Também é uma boa prática ter somente um único servidor JVM do WebSphere Liberty que esteja configurado em qualquer região única do CICS.

É possível hospedar o z/OS Connect for CICS 1.0 em sua própria região CICS, ou grupo de regiões CICS, e usar o mecanismo de Link de Programa Distribuído para chamar programas CICS nas regiões CICS que possuem o aplicativo.

Procedimento

1. Crie um JVMSERVER e configure-o para suportar o WebSphere Liberty. Para obter mais informações sobre como criar um JVMSERVER do WebSphere Liberty, consulte .
2. Configure o WebSphere Liberty para seus requisitos de segurança. Por padrão, o WebSphere Liberty espera o uso de certificados SSL certificados pelo cliente. Para ativar a Autenticação Básica HTTP, inclua a opção de configuração a seguir no arquivo `server.xml`:

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

Você também deve fornecer aos usuários do z/OS Connect a função de segurança **zosConnectAccess**. Para obter mais informações sobre a segurança do WebSphere Liberty, consulte Configurando a Segurança para um Servidor JVM do Liberty e para segurança do z/OS Connect, consulte Segurança para z/OS Connect.

3. Atualize a lista `<featureManager>` no arquivo `server.xml` para o ambiente do WebSphere Liberty para incluir um recurso `<feature>cicsts:zosConnect-1.0</feature>`, conforme mostrado no exemplo a seguir:

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

4. Defina o Controlador de Serviço do z/OS Connect for CICS 1.0, incluindo a instrução a seguir no arquivo `server.xml`:

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

5. Instale o JVMSERVER. Verifique o arquivo `messages.log` gerado em busca de mensagens de erro ou de aviso. Este log contém as mensagens que são geradas pelo WebSphere Liberty Server, incluindo mensagens que são retornadas pelo z/OS Connect for CICS 1.0, como as seguintes:

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

6. Crie um arquivo de configuração de pipeline XML. O arquivo de configuração de pipeline de amostra `jsonzosconnectprovider.xml` é fornecido no diretório `/usr/lpp/cicsts/cicsts54/samples/pipelines` (em que `/usr/lpp/cicsts/cicsts54` é o diretório de instalação padrão para arquivos CICS no z/OS UNIX). Você deve decidir se deseja analisar o JSON usando Java no servidor Liberty JVM (que é o padrão) ou usar o analisador JSON não Java:

- Para analisar o JSON usando Java no servidor Liberty JVM, é possível usar o arquivo de configuração de pipeline de amostra, mas substitua DFHWLP no elemento <jvmserver> pelo nome de seu JVMSERVER da Etapa 1.
- Para analisar o JSON usando o analisador não Java, modifique o arquivo de configuração de amostra para anexar o atributo java_parser="no" no elemento <provider_pipeline_json> como no exemplo a seguir:

```
<?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>
```

Substitua DFHWLP pelo nome do JVMSERVER que você criou no início deste procedimento.

7. Copie o arquivo de configuração de pipeline em um diretório adequado no zFS e assegure que as permissões do arquivo permitam que a região do CICS leia o arquivo. Para obter informações, consulte [Arquivos de configuração de pipeline](#).
8. Crie um recurso PIPELINE. O recurso PIPELINE define o local do arquivo de configuração de pipeline no atributo CONFIGFILE.
9. Opcional: Crie um recurso URIMAP padrão para z/OS Connect. Recursos URIMAP são usados para associar um TRANSACTION e o ID do usuário padrão ao serviço do z/OS Connect. Um ou mais recursos URIMAP podem ser usados para configurar uma política padrão para z/OS Connect. Para obter uma configuração de URIMAP de exemplo e mais informações sobre opções de configuração, consulte :

Nota:

O z/OS Connect executa autenticação extra para solicitações de HTTP individuais, portanto, as tarefas do aplicativo que são executadas no CICS normalmente estão associadas a um ID de usuário mais específico do que o ID do usuário inicial do URIMAP. O ID do usuário inicial fica em vigor somente até a autenticação específica do usuário acontecer dentro do z/OS Connect.

Resultados

Sua instância do z/OS Connect for CICS 1.0 é configurada. É possível testar a configuração básica digitando esta URL em um navegador da web: `https://hostname:portnumber/zosConnect/apim/services`, em que *hostname* é o endereço IP ou o nome do host do sistema no qual a região CICS que está hospedando z/OS Connect for CICS 1.0 está em execução e *portnumber* é a **httpsPort** que é especificada no elemento <httpEndpoint> do arquivo `server.xml`. O navegador da web exibe uma lista de serviços instalados; como nenhum serviço foi instalado ainda, a lista estará vazia.

Se você receber uma resposta HTTP 403 "AuthorizationFailed" em vez da lista de Serviços esperada, revise a configuração de Segurança da Etapa 2. É provável que o usuário autenticado não esteja autorizado a usar o z/OS Connect.

O que Fazer Depois

Agora você está pronto para implementar Serviços da Web JSON no z/OS Connect for CICS 1.0.

Configurando o z/OS Connect Enterprise Edition

z/OS Connect Enterprise Edition é um produto que pode ser solicitado separadamente; ele não é fornecido como parte do CICS TS. Primeiro, deve-se instalar o componente de tempo de execução do z/OS Connect Enterprise Edition e configurar um arquivo zFS para que o CICS possa localizá-lo. Em seguida, você configura um servidor JVM e define a configuração de pipeline e os recursos para o z/OS Connect, antes de poder implementar serviços JSON e APIs. Essa configuração inicial é uma atividade única que permite que o z/OS Connect Enterprise Edition seja executado integrado no servidor Liberty que é distribuído com o CICS TS.

Antes de Iniciar

Instale o tempo de execução do z/OS Connect Enterprise Edition. Siga as instruções no Documentação do z/OS Connect Enterprise Edition no IBM Knowledge Center, mas esteja ciente de que customizar o z/OS Connect Enterprise Edition para execução integrada dentro do CICS significa que você não precisa criar uma instância do servidor Liberty separada. Você só precisa instalá-lo até a extensão em que os componentes do sistema de arquivos estão disponíveis no zFS. A instalação do Editor de API que é fornecido com o z/OS Connect Enterprise Edition não faz parte desta tarefa.

Você instalou o z/OS Connect for CICS 1.0 anteriormente? Em caso afirmativo, várias das etapas nesta tarefa não são necessárias. As etapas que você pode ignorar são indicadas na lista.

Procedimento

1. Inclua ZCEE_INSTALL_DIR em suas opções do servidor JVM. Para obter mais informações sobre a opção e um exemplo, consulte Opções do servidor JVM.
2. Crie um JVMSERVER e configure-o para suportar o WebSphere Liberty. Para obter mais informações sobre como criar um JVMSERVER do WebSphere Liberty, consulte .

Se você instalou anteriormente o z/OS Connect for CICS 1.0, poderá ignorar esta etapa porque provavelmente já possui uma configuração adequada.

3. Configure o WebSphere Liberty para seus requisitos de segurança. Por padrão, ele espera o uso de certificados SSL certificados pelo cliente. Para ativar a Autenticação Básica HTTP, inclua a opção de configuração a seguir no arquivo `server.xml`:

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

Você também deve fornecer aos usuários do z/OS Connect a função de segurança `zosConnectAccess`. Para obter mais informações sobre a segurança do WebSphere Liberty, consulte Configurando a Segurança para um Servidor JVM do Liberty e para a segurança do z/OS Connect, consulte Segurança para z/OS Connect.

Se você instalou anteriormente o z/OS Connect for CICS 1.0, poderá ignorar esta etapa porque provavelmente já possui uma configuração adequada.

4. Atualize a lista `<featureManager>` no arquivo `server.xml` para o WebSphere Liberty para incluir um recurso `<feature>cicsts:zosConnect-2.0</feature>`, conforme mostrado no exemplo a seguir:

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


Nota: Os recursos z/OS Connect for CICS 1.0 e z/OS Connect Enterprise Edition são mutuamente exclusivos. Se você estiver migrando do z/OS Connect for CICS 1.0 para o z/OS Connect Enterprise Edition, mude o recurso `cicsts:zosConnect-1.0` existente para `cicsts:zosConnect-2.0`.

5. Defina o z/OS Connect Service Controller, incluindo a seguinte instrução no arquivo `server.xml`:

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

Se você instalou anteriormente o z/OS Connect for CICS 1.0, poderá ignorar esta etapa porque provavelmente já possui uma configuração adequada.

6. Instale o JVMSERVER. Verifique o arquivo `messages.log` gerado em busca de mensagens de erro ou de aviso. Este log contém as mensagens que são geradas pelo WebSphere Liberty Server, incluindo mensagens que são retornadas pelo z/OS Connect Enterprise Edition, conforme a seguir:

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

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

7. Crie um arquivo de configuração de pipeline XML. O arquivo de configuração de pipeline de amostra `jsonzosconnectprovider.xml` é fornecido no diretório `/usr/lpp/cicsts/cicsts54/samples/pipelines` (em que `/usr/lpp/cicsts/cicsts54` é o diretório de instalação padrão para arquivos CICS no z/OS UNIX). Você deve decidir se deseja analisar o JSON usando Java no servidor Liberty JVM (que é o padrão) ou usar o analisador JSON não Java:

- Para analisar o JSON usando Java no servidor Liberty JVM, é possível usar o arquivo de configuração de pipeline de amostra, mas substitua DFHWLP no elemento `<jvmserver>` pelo nome de seu JVMSERVER da Etapa 2.
- Para analisar o JSON usando o analisador não Java, modifique o arquivo de configuração de amostra para anexar o atributo `java_parser="no"` no elemento `<provider_pipeline_json>` como no exemplo a seguir:

```
<?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>
```

Substitua DFHWLP pelo nome do JVMSERVER que você criou no início deste procedimento.

Se você instalou anteriormente o z/OS Connect for CICS 1.0, poderá ignorar esta etapa porque provavelmente já possui uma configuração adequada.

8. Copie o arquivo de configuração de pipeline em um diretório adequado no zFS e assegure que as permissões do arquivo permitam que a região do CICS leia o arquivo. Para obter informações, consulte [Arquivos de configuração de pipeline](#).

Se você instalou anteriormente o z/OS Connect for CICS 1.0, poderá ignorar esta etapa porque provavelmente já possui uma configuração adequada.

9. Crie um recurso PIPELINE. O recurso PIPELINE define o local do arquivo de configuração de pipeline no atributo `CONFIGFILE`.

Se você instalou anteriormente o z/OS Connect for CICS 1.0, poderá ignorar esta etapa porque provavelmente já possui uma configuração adequada.

10. Opcional: Crie um recurso URIMAP padrão para o z/OS Connect. Os recursos URIMAP são usados para associar um TRANSACTION e ID do usuário padrão com o serviço do z/OS Connect. Um ou mais recursos URIMAP podem ser usados para configurar uma política padrão para o z/OS Connect.

Para obter um exemplo de configuração de URIMAP e mais informações sobre opções de configuração, consulte Configurando permissões para Serviços e APIs do z/OS Connect.

Nota:

O z/OS Connect executa a autenticação extra para solicitações de HTTP individuais, portanto, as tarefas do aplicativo que são executadas no CICS normalmente são associadas a um ID do Usuário mais específico do que o ID do Usuário inicial do URIMAP.

Esse ID do Usuário inicial fica em vigor somente até a autenticação específica do usuário ocorrer dentro do z/OS Connect. É recomendável especificar um ID do usuário específico, tal como ZOSCUSER no URIMAP, em vez de contar com que o ID do Usuário padrão do CICS seja usado (geralmente "CICSUSER") e autorizando que ele use as Transações de destino.

Resultados

Sua instância do z/OS Connect Enterprise Edition foi configurada. É possível testar a configuração básica para z/OS Connect digitando esta URL em um navegador da web: `https://hostname:portnumber/zosConnect/apim/services`, em que *hostname* é o endereço IP ou nome do host do sistema no qual a região CICS que está hospedando o z/OS Connect está em execução e *portnumber* é o **httpsPort** que é especificado no elemento `<httpEndpoint>` do arquivo `server.xml`. O navegador da web exibe uma lista de serviços instalados: esta lista está vazia, porque nenhum serviço foi instalado ainda.

Se você receber uma resposta HTTP 403 "AuthorizationFailed" em vez da lista Serviços esperada, revise a configuração de Segurança da Etapa 3. É provável que o usuário autenticado não esteja autorizado a usar o z/OS Connect.

O que Fazer Depois

Agora você está pronto para implementar serviços da web JSON ou APIs no z/OS Connect Enterprise Edition. Para obter informações sobre como implementar serviços, consulte Configurando o z/OS Connect para um serviço da web CICS JSON. Para obter informações sobre como implementar APIs, consulte "Usando as APIs do z/OS Connect Enterprise Edition" na página 78.

Configurando o z/OS Connect for CICS para um serviço da web CICS JSON

Depois de configurar inicialmente o z/OS Connect for CICS, você pode configurá-lo para um serviço da web CICS JSON. Os serviços da web JSON são implementados no z/OS Connect for CICS de um modo semelhante a outros ambientes de PIPELINE do CICS.

Antes de Iniciar

Você deve concluir a configuração básica do z/OS Connect antes de iniciar esta tarefa. Para z/OS Connect for CICS 1.0, consulte o "Configurando o z/OS Connect for CICS 1.0" na página 69. Para z/OS Connect Enterprise Edition, consulte o "Configurando o z/OS Connect Enterprise Edition" na página 72. Também é requerido um arquivo JSON WSBind para cada serviço que você deseja implementar; esses arquivos de ligação podem ser gerados usando o assistente CICS JSON (os programas utilitários **DFHLS2JS** e **DFHJS2LS**). Para obter mais

informações sobre esses programas utilitários, consulte .

Sobre Esta Tarefa

Os serviços da web JSON são implementados no z/OS Connect for CICS de um modo semelhante a como eles são implementados em outros ambientes de PIPELINE do CICS.

Procedimento

- Instale um recurso WEBSERVICE que associa o arquivo WSBind ao recurso PIPELINE apropriado. Selecione um nome significativo para o WEBSERVICE, pois este nome também é usado como o nome do Serviço no z/OS Connect. Opcionalmente, é possível usar o comando **PERFORM PIPELINE SCAN** para instalar recursos WEBSERVICE. Independentemente do método que você escolher usar, assegure-se de considerar as seguintes informações sobre URIMAPs:

O URI, no qual o Serviço está disponível no z/OS Connect, é obtido a partir um dos seguintes locais:

- A convenção de nomenclatura padrão que é usada pelo próprio z/OS Connect.
- O URI armazenado no arquivo WSBind, quando o recurso WEBSERVICE associado é instalado usando o comando **PERFORM PIPELINE SCAN**.
- A configuração específica de Serviço no arquivo `server.xml` do Liberty, se o parâmetro de configuração `invokeURI` é usado.
- Apenas z/OS Connect Enterprise Edition: o Application Archive File (arquivo `.aar`) para uma API que contém o Serviço.

Se você conta com a convenção de nomenclatura padrão, o Serviço normalmente é exposto por meio do URI a seguir:

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

Neste exemplo, `<Service Name>` refere-se ao nome do Serviço (mais especificamente, o nome do recurso WEBSERVICE) e `<hostname>` e `<port>` são obtidos da configuração do servidor Liberty.

- Opcional: Instale um recurso URIMAP para o Serviço do z/OS Connect. Um recurso URIMAP é usado pelo CICS para associar o trabalho para o Serviço a um ID de transação específico no CICS e a um ID do Usuário inicial. É possível usar um único URIMAP para configurar um ou vários Serviços. Você também pode definir um recurso URIMAP padrão ao configurar o CICS for z/OS Connect. Se nenhum URIMAP existir para corresponder ao URI para um Serviço do z/OS Connect, as tarefas do aplicativo serão executadas sob a transação CJSJA e o ID de usuário padrão do CICS (geralmente CICSUSER) será usado como o ID do usuário inicial. Para obter mais informações sobre a função do ID do Usuário inicial, consulte .

Se você optar por criar um recurso URIMAP, assegure que o ID do usuário associado possua autoridade para executar a transação especificada.

Você também pode optar por usar um URIMAP para associar o URI a um recurso WEBSERVICE específico. Se um URIMAP está estreitamente ligado a um WEBSERVICE, o WEBSERVICE de destino é usado para todas as solicitações de HTTP que são correspondidas pelo URIMAP. Se um WEBSERVICE não é especificado, o WEBSERVICE é selecionado com base no nome do Serviço do z/OS Connect. Se o atributo WEBSERVICE do URIMAP for deixado em branco, o z/OS Connect executará o mapeamento sozinho, o que permite que o z/OS Connect Enterprise Edition associe Serviços diferentes a diferentes métodos de HTTP em uma API.

- Opcional: Atualize o arquivo `server.xml`. Alguns padrões de uso podem exigir mudanças no arquivo de configuração do servidor Liberty, `server.xml`. Geralmente não é necessário fazer mudanças específicas do Serviço no arquivo `server.xml`, a menos que você requeira processamento de caso especial para um Serviço. Por exemplo, você pode optar por associar um conjunto específico de interceptores do z/OS Connect a um Serviço no arquivo `server.xml` ou expor um Serviço usando um URI que não corresponde à convenção de nomenclatura padrão do z/OS Connect.

Use essas etapas para definir um Serviço no arquivo `server.xml`:

1. Assegure que o nome do Serviço pretendido não corresponda ao nome de um recurso `WEBSERVICE` no CICS. O CICS injeta automaticamente informações de configuração no z/OS Connect e, se um Serviço definido explicitamente e um `WEBSERVICE` do CICS têm o mesmo nome, o comportamento resultante é imprevisível.
2. Configure um valor adequado para o atributo **`invokeURI`** no arquivo `server.xml` que corresponde ao URI usado no URIMAP.
3. Ligue o Serviço ao elemento `serviceRef` de **`CICSEndpointService`**.
4. Assegure que um URIMAP exista para corresponder o URI para o Serviço ao `WEBSERVICE` exato que deve ser usado. Se um atributo **`invokeURI`** estiver configurado na **Etapa b.**, o URIMAP deverá corresponder a esse URI. Caso contrário, o URIMAP deverá assumir a convenção de nomenclatura do URI padrão do z/OS Connect.

O exemplo a seguir mostra uma declaração de Serviço explícita do z/OS Connect for CICS 1.0 no arquivo `server.xml`:

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

O exemplo a seguir mostra a declaração equivalente para o z/OS Connect Enterprise Edition no arquivo `server.xml`:

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

Em ambos os exemplos, o URI `"/json/myCustomService"` é associado ao programa receptor do z/OS Connect para CICS, `CICSService1`.

Resultados

É possível testar a configuração para o z/OS Connect for CICS digitando esta URL em um navegador da web: `https://hostname:portnumber/zosConnect/apim/services`. O *hostname* é o endereço IP ou o nome da região CICS que hospeda o z/OS Connect for CICS. O *portnumber* é o **`httpsPort`** que está configurado na seção `<httpEndpoint>` do arquivo de configuração `server.xml`. O navegador da web exibe uma lista dos serviços instalados.

O serviço agora está pronto para ser chamado a partir de um cliente JSON que utiliza o mesmo *hostname* e *portnumber*.

Cada Serviço instalado possui um entrada na lista `zosConnect/apim/services`, por exemplo:

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

Neste exemplo, o recurso WEBSERVICE associado foi nomeado "EXAMPLE" e esta definição de Serviço foi criada dinamicamente pelo CICS quando o recurso WEBSERVICE EXAMPLE foi instalado. É possível usar um navegador da web para visitar <https://hostname:portnumber/zosConnect/apim/services/EXAMPLE> que retorna um documento com mais detalhes sobre o Serviço, semelhante ao seguinte:

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

Neste exemplo, o "address" é o URI no qual o serviço é exposto. Este URI pode ser derivado de informações em um arquivo WSBind, de um atributo **invokeURI** ou da convenção de nomenclatura padrão do z/OS Connect.

Quando uma solicitação chega no servidor Liberty JVM, ela é associada ao receptor do z/OS Connect for CICS que usa as informações do arquivo de configuração `server.xml`. Uma nova tarefa do CICS começa a executar este serviço e ela está associada a um recurso WEBSERVICE específico que usa as informações do recurso URIMAP. O processo de transformação de dados ocorre no servidor Liberty JVM e o programa CICS de destino está conectado, conforme denominado no arquivo WSBind.

Movendo serviços da web JSON de um Java Pipelines for JSON para z/OS Connect for CICS 1.0 ou z/OS Connect Enterprise Edition

O ambiente do z/OS Connect é compatível com a tecnologia Java Pipelines for JSON. É possível mover serviços da web JSON de um Java Pipelines for JSON para z/OS Connect for CICS 1.0 ou z/OS Connect Enterprise Edition.

Sobre Esta Tarefa

Por brevidade, nesta seção, "z/OS Connect" é usado para se referir ao z/OS Connect for CICS 1.0 ou ao z/OS Connect Enterprise Edition.

A reimplementação de um serviço da web JSON a partir de um Java Pipelines for JSON (conforme usado com o CICS Transaction Server Feature Pack for Mobile Extensions V1.0) para o z/OS Connect é um processo direto. Os arquivos WSBind que são usados para o Java Pipelines for JSON e para o z/OS Connect são produzidos usando as mesmas ferramentas (DFHLS2JS e DFHJS2LS) e são totalmente compatíveis entre si. Se você estiver explorando este recurso no CICS, poderá localizar exemplos de serviços da web JSON no Redbooks IBM: Implementando serviços da web CICS JSON para aplicativos móveis.

O ambiente do z/OS Connect encoraja o uso de SSL entre o aplicativo cliente e o CICS. Se o seu ambiente existente do Java Pipelines for JSON não usar SSL, a conversão para o z/OS Connect envolverá uma etapa extra.

Procedimento

1. Crie a infraestrutura necessária do z/OS Connect usando as instruções de Configurando o z/OS Connect for CICS. Para parte desta configuração, selecione um número da porta TCP/IP SSL na qual o servidor Liberty JVM atende conexões recebidas. Você tem duas opções para considerar:
 - a. Selecione um número de porta diferente do número da porta que é usado pelo TCPIPService para o Java Pipelines for JSON. Essa opção tem a vantagem de assegurar que ambos os ambientes possam ser instalados simultaneamente em portas TCP/IP diferentes. Significa que os programas clientes precisam de atualização para acessar o novo Serviço JSON. Se o ambiente antigo não usava SSL, a conversão para o z/OS Connect requer mudanças no URI, portanto, essa opção é mais adequada.
 - b. Selecione o mesmo número de porta que o usado pelo TCPIPService para o Java Pipelines for JSON. Os números de porta nos URIs usados pelos programas clientes não precisam ser mudados, mas os dois ambientes não podem ser instalados simultaneamente. O URI pode precisar mudar por outros motivos, tal como alternar de HTTP para HTTPS quando você ativa o SSL.
2. Implemente os arquivos WSBIND para o z/OS Connect. Seus arquivos WSBIND existentes são totalmente compatíveis com o z/OS Connect. Siga as etapas para implementar um novo serviço da web JSON para o z/OS Connect conforme descrito em Configurando o z/OS Connect para um serviço da web CICS JSON. O CICS não permite que dois recursos WEBSERVICE com o mesmo nome sejam instalados. Portanto, você tem duas opções:
 - a. Descarte o WEBSERVICE original para permitir que o novo seja instalado com o mesmo nome que foi usado anteriormente.
 - b. Renomeie o novo WEBSERVICE para evitar um conflito.
3. Se você customizou o processamento do Java Pipelines for JSON por meio do uso de programas PIPELINE Handler, considere se essa customização ainda é necessária. Se ela for necessária, crie programas do z/OS Connect Interceptor com funções equivalentes e implemente-os como interceptores globais. Para obter mais informações sobre os interceptores do z/OS Connect, consulte Definindo interceptores do z/OS Connect.

Resultados

Você está pronto para testar seus novos serviços da web JSON do z/OS Connect. Se você implementou os serviços usando o mesmo número de porta usada no Java Pipelines for JSON, nenhuma mudança é necessária no cliente (a menos que a configuração de segurança tenha mudado, tal como ao ativar o SSL). Se você mudou o número da porta ou o URI para o serviço, o cliente precisa mudar.

Usando as APIs do z/OS Connect Enterprise Edition

Um recurso principal do z/OS Connect Enterprise Edition é a capacidade de compor APIs de JSON a partir de um ou mais serviços JSON. Para trabalhar com APIs, deve-se instalar o componente Editor de API do z/OS Connect Enterprise Edition.

Para obter detalhes sobre o Editor de API, consulte Documentação do z/OS Connect Enterprise Edition no IBM Knowledge Center.

Existem algumas pequenas diferenças na implementação de APIs no z/OS Connect Enterprise Edition para CICS, em comparação com a implementação de APIs em uma instalação independente do z/OS Connect Enterprise Edition. As seguintes considerações se aplicam aos implementar APIs para CICS:

- Não é possível adotar um arquivo WSBIND existente para uso com uma API. O Editor de API requer um ou mais arquivos de Service Archive Resource (SAR) como parte de sua entrada. Este arquivo é criado com os assistentes BAQLS2JS ou BAQJS2LS que são distribuídos com o z/OS Connect Enterprise Edition. Se você deseja criar APIs, comece gerando quaisquer arquivos WSBIND e SAR necessários usando os assistentes que são fornecidos com o z/OS Connect Enterprise Edition.

Deve-se configurar o parâmetro **SERVICE-NAME** em BAQLS2JS ou BAQJS2LS para corresponder ao nome do recurso WEBSERVICE que será usado no CICS. O CICS depende de uma correspondência de um para um entre o nome do Serviço no z/OS Connect e o nome do WEBSERVICE no CICS.

- É necessária uma definição de recurso WEBSERVICE no CICS que corresponda ao nome do Serviço. O WEBSERVICE encapsula o arquivo WSBIND para o Serviço.

- Uma definição de recurso URIMAP também pode ser fornecida no CICS. Se um URIMAP for usado, o CICS associará automaticamente o serviço para a API com um ID de transação e configurará um ID do usuário inicial adequado.

- O Editor de API produz um arquivo Application Archive Resource (AAR) como saída. Isso deve ser implementado no servidor Liberty usando os mecanismos de implementação fornecidos pelo z/OS Connect Enterprise Edition. Pode ser necessário criar um diretório adequado para as APIs na área de trabalho do servidor Liberty. As APIs são implementadas no diretório /resources/zosconnect/apis na área de trabalho do Liberty. Se esse diretório ainda não existir, crie-o. Um exemplo de uso de apideploy é o seguinte:

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

No exemplo fornecido, /tmp/ExampleAPI.aar é o local do arquivo Application Archive Resource que está sendo implementado e /u/userid/APPLID/JVMSEVER/wlp/usr/servers/ é o valor da propriedade WLP_OUTPUT_DIR no arquivo de configuração JVMSEVER. Este exemplo resulta em apideploy atualizando a configuração do servidor Liberty que está sendo usado pelo CICS.

Se você seguir as considerações acima, a API e seus serviços do componente serão implementados no CICS. O CICS está configurado com recursos URIMAP e WEBSERVICE adequados e o arquivo AAR é implementado na área de trabalho de configuração do Liberty. O arquivo de configuração principal do Liberty, server.xml, contém uma entrada para cada serviço de componente que é implementado.

Arquivos de configuração de pipeline

A configuração de um pipeline utilizado para manipular uma solicitação de serviço da web é especificada em um documento XML, conhecido como *arquivo de configuração do pipeline*.

O arquivo de configuração de pipeline é armazenado no sistema de arquivos do z/OS UNIX System Services e seu nome é especificado no atributo CONFIGFILE de uma definição de recurso PIPELINE. Utilize um editor de texto ou editor XML adequado para trabalhar com arquivos de configuração de pipeline. Os esquemas

XML para os arquivos de configuração de pipeline estão no diretório `/usr/lpp/cicsts/cicsts54/schemas/pipeline/` (em que `/usr/lpp/cicsts/cicsts54` é o diretório de instalação padrão para arquivos do CICS no z/OS UNIX). Ao trabalhar com arquivos de configuração, assegure-se de que a codificação do conjunto de caracteres seja UTF-8. Se você importar um arquivo de configuração existente que esteja codificado em EBCDIC, ele será automaticamente convertido em UTF-8.

Quando o CICS processa uma solicitação de serviço da web, ele usa um pipeline de um ou mais manipuladores de mensagem para manipular a solicitação. Um pipeline é configurado para fornecer aspectos do ambiente de execução que se aplicam a diferentes categorias de aplicativos, tal como o suporte para segurança de serviço da web e transações de serviço da web. Geralmente, uma região CICS que possui um grande número de aplicativos do provedor de serviços ou do solicitante de serviço precisa de várias configurações de pipeline diferentes. No entanto, onde aplicativos diferentes possuem requisitos semelhantes, eles podem compartilhar a mesma configuração de pipeline.

Nota: Ao usar o CICS Explorer para criar um novo arquivo de configuração de PIPELINE como parte de um pacote configurável, não deve haver um arquivo de configuração com o mesmo nome na raiz do pacote configurável.

Há dois tipos de configurações de pipeline: um descreve a configuração de um pipeline do provedor de serviços; o outro descreve um pipeline do solicitante de serviço. Cada um é definido por seu próprio esquema e cada um possui um elemento-raiz diferente.

Em Processo	Schema	Elemento Raiz
Provedor de Serviços	Provider.xsd	<provider_pipeline>
Solicitante de serviços	Requester.xsd	<requester_pipeline>

Embora muitos dos elementos XML utilizados sejam comuns para ambos os tipos de configuração de pipeline, outros são utilizados somente em um ou no outro, portanto, você não pode utilizar o mesmo arquivo de configuração para um provedor e um solicitante.

Restrição: Nomes de elemento qualificados pelo namespace não são suportados no arquivo de configuração de pipeline.

Os elementos <provider_pipeline> e <requester_pipeline> possuem os seguintes subelementos imediatos:

- Um elemento <service>, que especifica os manipuladores de mensagem que são chamados para cada solicitação. Este elemento é obrigatório quando usado dentro do elemento <provider_pipeline> e opcional dentro do elemento <requester_pipeline>.
- Um elemento <transport> opcional, que especifica os manipuladores de mensagens que são selecionados no tempo de execução, com base nos recursos que estão sendo utilizados para o transporte de mensagens.
- Apenas para <provider_pipeline>, um elemento <apphandler> opcional, que é usado para especificar manipuladores de aplicativo conectados ao canal.
- Apenas para <provider_pipeline>, um elemento <apphandler_class> opcional, que é usado para especificar um manipulador de aplicativo Axis2.
- Um elemento <service_parameter_list> opcional, que contém os parâmetros que estão disponíveis para os manipuladores de mensagem no pipeline.

Determinados elementos podem ter atributos associados a eles. Cada valor de atributo deve ter aspas ao redor dele para produzir um documento XML válido.

Associado ao arquivo de configuração de pipeline há um recurso PIPELINE. Os atributos incluem CONFIGFILE, que especificar o nome do arquivo de configuração de pipeline no z/OS UNIX. Ao instalar uma definição de PIPELINE, o CICS lê as informações de que ele precisa para configurar o pipeline a partir do arquivo.

O CICS fornece arquivos de configuração de amostra que podem ser usados como uma base para o desenvolvimento de seus próprios arquivos de configuração. Eles são fornecidos na biblioteca /usr/lpp/cicsts/cicsts54/samples/pipelines.

basicsoap11provider.xml

Uma definição de pipeline do provedor de serviços que usa o protocolo SOAP 1.1 para um pipeline que não suporta Java. O pipeline usa o manipulador de mensagem <cics_soap_1.1_handler> e é usado quando o aplicativo CICS tiver sido implementado usando o assistente de serviços da web do CICS.

basicsoap11requester.xml

Uma definição de pipeline do solicitante de serviço que usa o protocolo SOAP 1.1 para um pipeline que não suporta Java. O pipeline usa o manipulador de mensagem <cics_soap_1.1_handler> e é usado quando o aplicativo CICS tiver sido implementado usando o assistente de serviços da web do CICS.

basicsoap11javaprovider.xml

Uma definição de pipeline do provedor de serviços que usa o protocolo SOAP 1.1 para um pipeline que suporta Java. O pipeline usa o manipulador de mensagem <cics_soap_1.1_handler_java> e é usado quando o aplicativo foi implementado usando o assistente de serviços da web CICS. Esta configuração contém o elemento <jvmserver>. Este manipulador de mensagem deve ser editado para especificar o servidor JVM apropriado antes que a configuração possa ser usada.

basicsoap11javarequester.xml

Uma definição de pipeline do solicitante de serviço que usa o protocolo SOAP 1.1 para um pipeline que suporta Java. O pipeline usa o manipulador de mensagem <cics_soap_1.1_handler_java> e é usado quando o aplicativo foi implementado usando o assistente de serviços da web CICS. Esta configuração contém o elemento <jvmserver>. Este manipulador de mensagem deve ser editado para especificar o servidor JVM apropriado antes que a configuração possa ser usada.

jsonjavaprovider.xml

Uma definição de pipeline do provedor de serviços que usa o formato da mensagem JSON para um pipeline que suporta Java. O pipeline usa o manipulador de mensagem <cics_json_handler_java> e é usado quando o aplicativo do CICS tiver sido implementado usando o assistente do CICS JSON. Esta configuração contém o elemento <jvmserver>. Este manipulador de mensagem deve ser editado para especificar o servidor JVM apropriado antes que a configuração possa ser usada.

jsonzosconnectprovider.xml

Uma definição de pipeline para um serviço da web JSON que é implementado em um PIPELINE que está configurado para z/OS Connect for CICS. O pipeline usa o manipulador de mensagem <provider_pipeline_json>. Esta configuração contém o elemento

<jvmserver>. Este manipulador de mensagem deve ser editado para especificar o servidor JVM apropriado antes que a configuração possa ser usada.

kerberosprovider.xml

Uma definição de pipeline do provedor de serviços que inclui informações de configuração para o suporte do Kerberos no basicsoap11provider.xml.

samlprovider.xml

Uma definição de pipeline do provedor de serviços que inclui informações de configuração para o suporte SAML no basicsoap11provider.xml.

samlrequester.xml

Uma definição de pipeline do solicitante de serviço que inclui informações de configuração para o suporte SAML no basicsoap11requester.xml.

propagatesamlprovider.xml

Uma definição de pipeline do provedor de serviços que inclui informações de configuração para suporte SAML com a propagação de informações de SAML por meio de uma transação do CICS para basicsoap11provider.xml.

propagatesamlrequester.xml

Uma definição de pipeline do solicitante de serviço que inclui informações de configuração para suporte SAML com a propagação de informações de SAML por meio de uma transação do CICS para basicsoap11requester.xml.

wsatprovider.xml

Uma definição de pipeline que inclui informações de configuração para transações de serviço da web para basicsoap11provider.xml.

wsatrequester.xml

Uma definição de pipeline que inclui informações de configuração para transações de serviço da web para basicsoap11requester.xml.

Arquivo de configuração de pipeline do provedor de exemplo (manipulador de aplicativo JSON)

Este é um exemplo simples de um arquivo de configuração para um pipeline do provedor de serviços que usa o 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>
```

O pipeline contém somente um manipulador de mensagem. O manipulador vincula ao programa DFHJSON.

- O elemento <provider_pipeline> é o elemento-raiz do arquivo de configuração de pipeline para um pipeline do provedor de serviços.
- O elemento <service> especifica os manipuladores de mensagem que são chamados para cada solicitação. Neste exemplo, há somente um manipulador de mensagem.

- O elemento <terminal_handler> contém a definição do manipulador de mensagem do terminal do pipeline.
- O elemento <cics_json_handler_java> indica que o pipeline é um pipeline baseado em Java e o manipulador de serviço do pipeline é um manipulador de mensagem que suporta mensagens JSON.
- O elemento <apphandler> especifica o nome do manipulador de aplicativo ao qual o manipulador de terminal do pipeline se vincula por padrão. Neste caso, o programa é DFHJSON, que é o programa fornecido pelo CICS para aplicativos implementados com o assistente CICS JSON.

Arquivo de Configuração de Pipeline do Provedor de Exemplo (Manipulador de Aplicativo Conectado ao Canal)

Este é um exemplo simples de um arquivo de configuração para um pipeline do provedor de serviços que usa o elemento <cics_soap_1.1_handler>:

```
<?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/>
    </terminal_handler>
  </service>
  <apphandler>DFHPITP</apphandler>
</provider_pipeline>
```

O pipeline contém somente um manipulador de mensagem. O manipulador vincula ao programa DFHPITP.

- O elemento <provider_pipeline> é o elemento-raiz do arquivo de configuração de pipeline para um pipeline do provedor de serviços.
- O elemento <service> especifica os manipuladores de mensagem que são chamados para cada solicitação. Neste exemplo, há somente um manipulador de mensagem.
- O elemento <terminal_handler> contém a definição do manipulador de mensagem do terminal do pipeline.
- O elemento <cics_soap_1.1_handler> indica que o pipeline não é um pipeline baseado em Java e o manipulador de terminal do pipeline é um manipulador de mensagem que suporta mensagens SOAP 1.1.
- O elemento <apphandler> especifica o nome do manipulador de aplicativo ao qual o manipulador de terminal do pipeline se vincula por padrão. Neste caso, o programa é DFHPITP, que é o programa fornecido pelo CICS para aplicativos implementados com o assistente de serviços da web CICS.

Arquivo de Configuração do Pipeline do Provedor de Exemplo (Manipulador de Aplicativo Axis2)

Este é um exemplo simples de um arquivo de configuração para um pipeline do provedor de serviços que usa o 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>
```

O pipeline contém somente um manipulador de mensagem. O manipulador vincula ao programa DFHPITP.

- O elemento `<provider_pipeline>` é o elemento-raiz do arquivo de configuração de pipeline para um pipeline do provedor de serviços.
- O elemento `<service>` especifica os manipuladores de mensagem que são chamados para cada solicitação. Neste exemplo, há somente um manipulador de mensagem.
- O elemento `<terminal_handler>` contém a definição do manipulador de mensagem do terminal do pipeline.
- O elemento `<cics_soap_1.1_handler_java>` indica que o pipeline é um pipeline baseado em Java e o manipulador de serviço do pipeline é um manipulador de mensagem que suporta mensagens SOAP 1.1.
- O elemento `<apphandler_class>` especifica o manipulador de aplicativo Axis2 fornecido.

Arquivo de Configuração do Pipeline do Solicitante de Exemplo

Este é um exemplo simples de um arquivo de configuração para um pipeline do solicitante de serviço que usa o elemento `<cics_soap_1.2_handler_java>` com suporte Axis2 MTOM/XOP:

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

O pipeline contém somente um manipulador de mensagem.

- O elemento `<requester_pipeline>` é o elemento-raiz do arquivo de configuração de pipeline para um pipeline do solicitante de serviço.
- O elemento `<service>` especifica os manipuladores de mensagem que são chamados para cada solicitação. Neste exemplo, há somente um manipulador de mensagem.
- O `<service_handler_list>` especifica uma lista de manipuladores de mensagem que são chamados para cada solicitação.
- O elemento `<cics_soap_1.2_handler_java>` indica que o pipeline suporta Java e o manipulador de serviço do pipeline é um manipulador de mensagem que suporta mensagens SOAP 1.2.
- O elemento `<jvmserver>` especifica o servidor da JVM a ser usado.
- O elemento `<mtom/>` especifica que documentos XOP de saída são empacotados em mensagens MTOM e enviados. Por padrão, mensagens MTOM de entrada são aceitas e descompactadas para pipelines baseados em Java.

Arquivo de configuração de pipeline do provedor de exemplo para um serviço da web JSON do z/OS Connect for CICS

Este é um exemplo simples de um arquivo de configuração para um pipeline do provedor de serviços que usa o elemento `<provider_pipeline_json>`. Como um atributo `java_parser="N0"` é fornecido, ele usa o analisador JSON não Java:

```
<?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>
```

O elemento <provider_pipeline_json> difere do elemento <provider_pipeline> pelo fato de que os programas do manipulador não podem ser definidos.

- O elemento <provider_pipeline_json> é o elemento raiz do arquivo de configuração de pipeline para um pipeline do provedor de serviços da web JSON do z/OS Connect for CICS.
- O atributo java_parser="NO" especifica que o analisador JSON não Java seja utilizado.
- O elemento <jvmserver> especifica o servidor da JVM a ser usado.

Nota: Uma tentativa de iniciar um pipeline <provider_pipeline_json> usando algo diferente do z/OS Connect for CICS resulta em um erro.

Arquivo de configuração de pipeline do provedor de exemplo para análise sintática de JSON não 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>
```

O pipeline contém somente um manipulador de mensagem.

- O elemento <provider_pipeline> é o elemento-raiz do arquivo de configuração de pipeline para um pipeline do provedor de serviços.
- O elemento <service> especifica os manipuladores de mensagem que são chamados para cada solicitação. No exemplo, há somente um manipulador de mensagem.
- O elemento <terminal_handler> contém a definição do manipulador de mensagem do terminal do pipeline.
- O elemento <handler> especifica detalhes do manipulador.
- O elemento <program> especifica o programa a ser chamado. DFHPIJT é o manipulador fornecido pelo CICS para processamento JSON não Java.

Manipuladores Relacionados ao Transporte

No arquivo de configuração para cada pipeline, você pode especificar mais de um conjunto de manipuladores de mensagens. No tempo de execução, o CICS seleciona os manipuladores de mensagens que são chamados, com base nos recursos que estão sendo utilizados para o transporte de mensagens.

Em um provedor de serviços, e em um solicitante de serviços, é possível especificar que alguns manipuladores de mensagens devem ser chamados somente quando um transporte específico (HTTP ou WebSphere MQ) está em uso. Por exemplo, considere um serviço da web que você torna disponível aos seus funcionários. As pessoas que trabalham em um local da empresa acessam o serviço utilizando o transporte do WebSphere MQ em uma rede interna segura; no entanto, os funcionários que trabalham em um local do parceiro de negócios acessam o serviço utilizando o transporte HTTP por meio da Internet. Nessa

situação, você pode desejar utilizar manipuladores de mensagem para criptografar partes da mensagem quando o transporte HTTP é utilizado, devido à natureza sensível das informações.

Em um provedor de serviços, as mensagens de entrada são associadas a um recurso nomeado (um TCPIPSERVICE para o transporte HTTP, um QUEUE para o transporte do MQ). É possível especificar que alguns manipuladores de mensagens devem ser chamados somente quando um recurso específico é utilizado para uma solicitação de entrada.

Para tornar isso possível, os manipuladores de mensagens são especificadas em duas partes distintas do arquivo de configuração de pipeline:

A seção de serviço

Especifica os manipuladores de mensagem que são chamados toda vez que o pipeline é executado.

A seção de transporte

Especifica os manipuladores de mensagem que podem ou não ser chamados, dependendo dos recursos de transporte que estão em uso.

Lembre-se: No tempo de execução, um manipulador de mensagem pode escolher restringir a execução do pipeline. Portanto, mesmo se o CICS decidir que um manipulador de mensagem específico deve ser chamado com base em qual está no arquivo de configuração de pipeline, a decisão pode ser indeferida por um manipulador de mensagem anterior.

Os manipuladores de mensagem que são especificados dentro da seção de transporte (os *manipuladores relacionados ao transporte*) são organizados em várias listas. No tempo de execução, o CICS seleciona os manipuladores somente em uma destas listas para execução, com base em quais recursos de transporte estão em uso. Se mais de uma lista corresponder aos recursos de transporte que estão sendo usados, o CICS usará a lista que é mais seletiva. As listas que são usadas nos pipelines do provedor de serviços e do solicitante de serviço são:

<default_transport_handler_list>

Esta é a lista menos seletiva de manipuladores relacionados ao transporte; os manipuladores especificados nesta lista são chamados quando nenhuma das listas a seguir corresponde aos recursos de transporte que estão sendo usados.

<default_http_transport_handler_list>

Em um pipeline do solicitante de serviço, os manipuladores nesta lista são chamados quando o transporte HTTP está em uso.

Em um pipeline do provedor de serviços, os manipuladores nesta lista são chamados quando o transporte HTTP está em uso e nenhum <named_transport_entry> nomeia o TCPIPSERVICE para a conexão TCP/IP.

<default_mq_transport_handler_list>

Em um pipeline do solicitante de serviço, os manipuladores nesta lista são chamados quando o transporte do WebSphere MQ está em uso.

Em um pipeline do provedor de serviços, os manipuladores nesta lista são chamados quando o transporte do WebSphere MQ está em uso e nenhum <named_transport_entry> nomeia a fila de mensagens na qual mensagens de entrada são recebidas.

A lista a seguir de manipuladores de mensagem é usada somente no arquivo de configuração para um pipeline do provedor de serviços:

<named_transport_entry>

Assim como uma lista de manipuladores, o <named_transport_entry> especifica o nome de um recurso e o tipo de transporte.

- Para o transporte HTTP, os manipuladores nesta lista são chamados quando o nome do recurso corresponde ao nome do TCPIPService para a conexão TCP/IP de entrada.
- Para o transporte WebSphere MQ, os manipuladores nesta lista são chamados quando o nome do recurso corresponde ao nome da fila de mensagens que recebe a mensagem de entrada.

Exemplo

Este é um exemplo de um elemento <transport> a partir do arquivo de configuração de pipeline para um pipeline do provedor de serviços:

```
<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 TCPIPService(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>
```

O efeito dessa definição é este:

- <default_mq_transport_handler_list> assegura que mensagens que usam o transporte do MQ sejam processadas pelo manipulador HANDLER3.
- O <named_transport_entry> assegura que mensagens que usam a conexão TCP/IP associada ao TCPIPService (WS00) sejam processadas pelo manipulador HANDLER4.
- <default_transport_handler_list> assegura que todas as mensagens restantes, ou seja, aquelas que usam o transporte HTTP, mas não TCPIPService(WS00), sejam processadas pelos manipuladores HANDLER1 e HANDLER2.

Lembre-se: Quaisquer manipuladores especificados na seção de serviço da definição de pipeline serão chamados além daqueles especificados na seção de transporte.

A Definição de Pipeline para um Provedor de Serviços

Os manipuladores de mensagem são definidos em um documento XML, que é armazenado no z/OS UNIX. O nome do arquivo que contém o documento é especificado no atributo CFGFILE de uma definição de PIPELINE.

O elemento-raiz do documento de configuração de pipeline é o elemento <provider_pipeline>. A estrutura de alto nível do documento é mostrada em Figura 20.

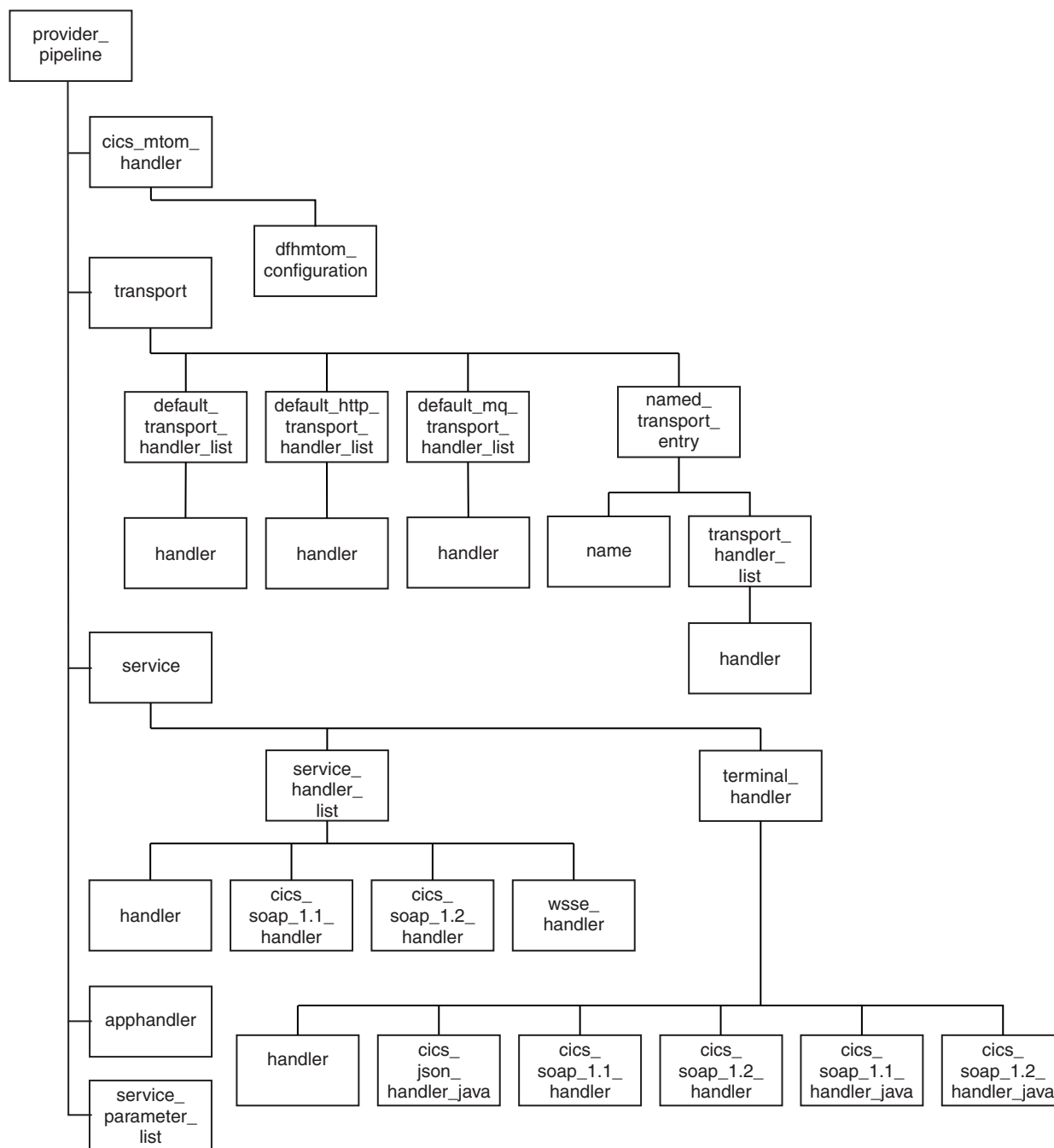


Figura 20. Estrutura da Definição de Pipeline para um Provedor de Serviços.

Nota: Para simplificar a figura, elementos filhos dos elementos <handler>, <cics_json_handler_java>, <cics_soap_1.1_handler>, <cics_soap_1.2_handler>, <cics_soap_1.1_handler_java> e <cics_soap_1.2_handler_java> não são mostrados.

A Definição de Pipeline para um Solicitante de Serviço

Os manipuladores de mensagem são definidos em um documento XML, o qual é armazenado no z/OS UNIX. O nome do arquivo que contém o documento é especificado no atributo CFGFILE de uma definição de PIPELINE.

O elemento-raiz do documento de configuração de pipeline é o elemento <requester_pipeline>. A estrutura de alto nível do documento é mostrada em Figura 21.

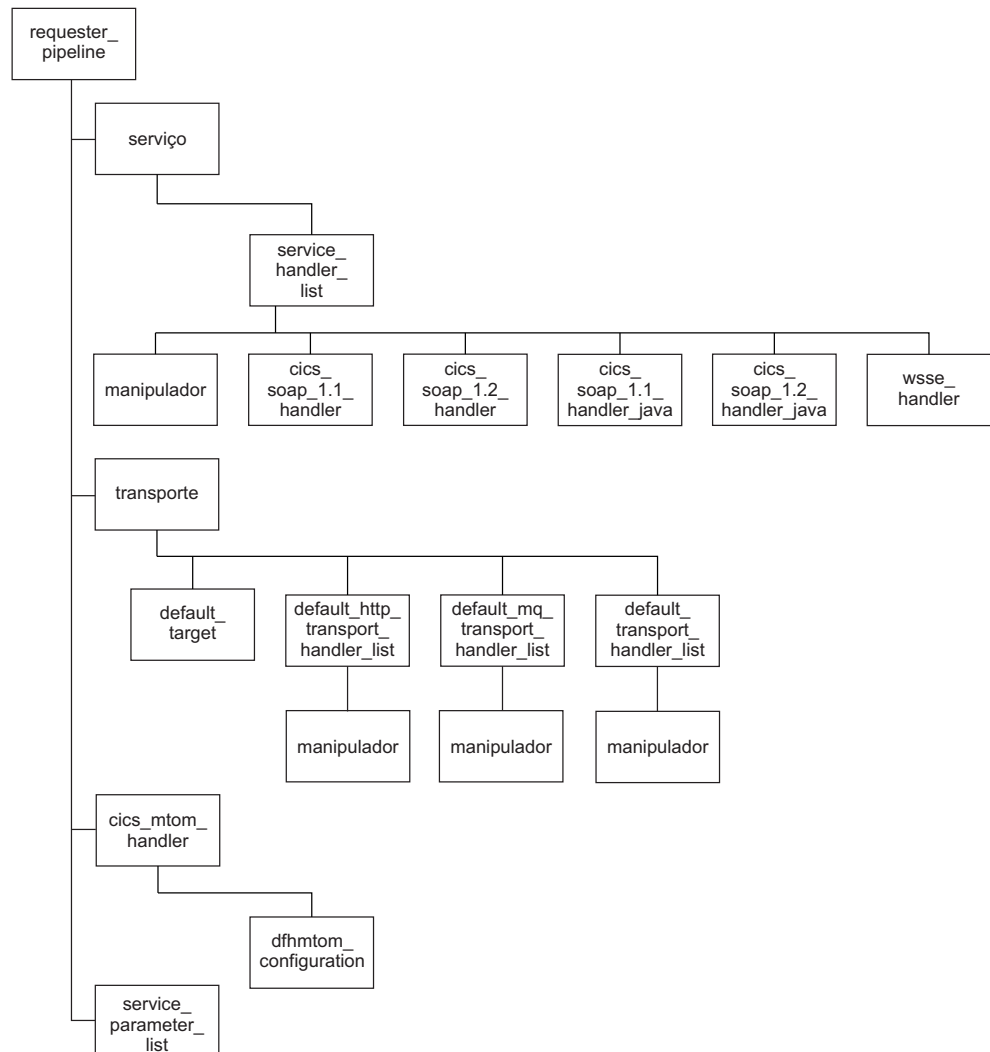


Figura 21. Estrutura da Definição de Pipeline para um Solicitante de Serviço.

Nota: Para simplificar a figura, elementos filhos dos elementos <handler>, <cics_soap_1.1_handler>, <cics_soap_1.2_handler>, <cics_soap_1.1_handler_java>, e <cics_soap_1.2_handler_java> não são mostrados.

Elementos Usados Apenas em Provedores de Serviços

Alguns dos elementos XML usados em um arquivo de configuração de pipeline se aplicam somente aos pipelines do provedor de serviços.

Manipuladores de Aplicativo:

Um manipulador de aplicativo é um programa CICS que o manipulador de terminal de um pipeline do provedor de serviços SOAP vincula ao tempo de execução.

Os manipuladores de aplicativo são usados em pipelines do modo de provedor nos quais o manipulador de terminal é um dos manipuladores de mensagem SOAP fornecidos. Esta situação ocorre quando o elemento `<terminal_handler>` contém um elemento `<cics_soap_1.1_handler>`, `<cics_soap_1.2_handler>`, `<cics_soap_1.1_handler_java>` ou `<cics_soap_1.2_handler_java>`.

O manipulador de aplicativo é responsável por processar o corpo de uma solicitação SOAP e por gerar uma resposta usando os dados retornados. O manipulador de aplicativo pode chamar outros programas para concluir este processamento. Geralmente o manipulador de aplicativo age como uma camada de apresentação de propósito geral em torno de um ou mais aplicativos de negócios. Ele é responsável por mapear XML para um formulário que um aplicativo pode usar, conectando esse aplicativo e, em seguida, gerando uma resposta usando os dados retornados.

Um manipulador de aplicativo pode ser conectado pelo CICS de duas maneiras. O mecanismo típico envolve contêineres de canal e de controle; o outro método envolve ligações Java para Axis2.

Os manipuladores de aplicativo conectados ao canal são especificados no elemento `<apphandler>` do elemento `<provider_pipeline>`. No tempo de execução, o contêiner DFHWS-APPHANDLER é preenchido pelo conteúdo de `<apphandler>`. No entanto, o contêiner DFHWS-APPHANDLER pode ser atualizado dinamicamente por qualquer um dos outros manipuladores de mensagem. Portanto, o programa que é vinculado ao tempo de execução pode ser diferente para o programa especificado no elemento `<apphandler>`. Os manipuladores de aplicativo a seguir podem ser especificados no elemento `<apphandler>` ou no contêiner DFHWS-APPHANDLER:

- O manipulador de aplicativo SOAP conectado ao canal fornecido, DFHPITP. Para obter mais informações sobre manipuladores de aplicativo conectados ao canal, consulte “Manipuladores de Aplicativo Conectados ao Canal” na página 133
- Seu próprio manipulador de aplicativo conectado ao canal. Este manipulador de aplicativo pode ser gravado em idiomas diferentes de Java. Para obter mais informações sobre os contêineres de controle que podem ser usados em seu manipulador de aplicativo conectado ao canal, consulte “Contêineres de Controle” na página 150.
- Seu próprio manipulador de aplicativo Java para pipelines baseados em Java, que implementa a interface Java `ApplicationHandler` e que está conectado ao pipeline usando `MessageContext` do Axis2. Para obter mais informações sobre a interface Java `ApplicationHandler`, consulte Referência de Classe JCICS.

Para usar um manipulador de aplicativo que usa ligações Java para Axis2, você deve especificar o elemento `<apphandler_class>` do elemento `<provider_pipeline>`. Os manipuladores de aplicativo Axis2 também requerem que um servidor de JVM exista para o pipeline de serviços da web e o manipulador de aplicativo para ser executado e que o manipulador terminal de seu pipeline de serviços da web deve ser o manipulador de mensagem `<cics_soap_1.1_handler_java>` ou `<cics_soap_1.2_handler_java>`. Para usar o manipulador de aplicativo Axis2 fornecido, você deve especificar

`com.ibm.cicsts.axis2.CICSAxis2ApplicationHandler` no elemento `<apphandler_class>`, no entanto, é possível especificar sua própria classe do manipulador de aplicativo Axis2. No tempo de execução, o contêiner DFHWS-APPHANCLAS é preenchido pelo conteúdo de `<apphandler_class>`.

Para aplicativos de serviço da web que são implementados usando o assistente de serviços da web do CICS, você deve especificar DFHPITP ou seu próprio manipulador de aplicativo que usa DFHPITP no elemento `<apphandler>` ou especificar `com.ibm.cicsts.axis2.CICSAxis2ApplicationHandler` no elemento `<apphandler_class>`. Para obter mais informações sobre o assistente de serviços da web do CICS, consulte O Assistente de Serviços da Web CICS.

Também é possível implementar aplicativos Axis2 como serviços da web no modo de provedor no CICS usando o estilo Axis2 de implementação de serviço da web. Para obter mais informações, consulte Implementando um serviço da web no modo de provedor Java em um servidor JVM Axis2.

O Elemento `<apphandler_class>`:

Especifica que o manipulador do terminal do pipeline vincula-se a um manipulador de aplicativo Axis2.

O elemento `<apphandler_class>` é usado para especificar um manipulador de aplicativo Axis2 quando seu elemento `<terminal_handler>` contém um elemento `<cics_json_handler_java>`, `<cics_soap_1.1_handler_java>` ou `<cics_soap_1.2_handler_java>`. Para usar o manipulador de aplicativo Axis2 fornecido, especifique `com.ibm.cicsts.axis2.CICSAxis2ApplicationHandler` no elemento `<apphandler_class>`. Ao usar os manipuladores CICS SOAP, também é possível especificar sua própria classe do manipulador de aplicativo Axis2.

Alternativamente, é possível especificar o elemento `<apphandler>` em seu arquivo de configuração de pipeline se desejar usar um manipulador de aplicativo conectado ao canal, para obter mais informações, consulte o elemento `<apphandler>`. Entretanto, você não deve especificar os elementos `<apphandler_class>` e `<apphandler>` no mesmo arquivo de configuração de pipeline.

Nota: Você não deve usar o elemento `<apphandler>` com o elemento `<cics_json_handler_java>`.

Você não deve usar o elemento `<apphandler_class>` se seu elemento `<terminal_handler>` contiver um elemento `<cics_soap_1.1_handler>` ou `<cics_soap_1.2_handler>`.

Para obter mais informações sobre manipuladores de aplicativo, consulte “Manipuladores de Aplicativo” na página 90.

Utilizado em:

- Provedor de Serviços

Contido por:

- Elemento `<provider_pipeline>`

Exemplo

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

O Elemento `<named_transport_entry>`:

Contém uma lista de manipuladores que devem ser chamados quando um recurso de transporte nomeado está sendo usado por um provedor de serviços.

- Para o transporte do WebSphere MQ, o recurso nomeado é a fila de entrada local na qual a solicitação é recebida.
- Para o transporte HTTP, o recurso é o TCPIPService que define a porta na qual a solicitação foi recebida.

Utilizado em:

- Provedor de Serviços

Contido por:

`<transport>`

Atributos:

Nome	Descrição
type	O mecanismo de transporte com o qual o recurso nomeado está associado: wmq O recurso nomeado é uma fila http O recurso nomeado é um TCPIPService

Contém:

1. Um elemento `<name>`, contendo o nome do recurso
2. Um elemento `<transport_handler_list>` opcional. Cada `<transport_handler_list>` contém um ou mais elementos `<handler>`.
Se você não codificar um elemento `<transport_handler_list>`, os únicos manipuladores de mensagem que serão chamados quando o transporte nomeado for usado serão aqueles especificados no elemento `<service>`.

Exemplo

```
<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>
```

Neste exemplo, os manipuladores de mensagem especificados (HANDLER1 e HANDLER2) são chamados para mensagens recebidas no TCPIPService com o nome PORT80.

O elemento `<provider_pipeline>`:

Especifica o elemento-raiz do documento XML que descreve a configuração do pipeline do CICS para um provedor de serviço da web.

Utilizado em:

- Provedor de Serviços

Contém:

1. Elemento <cics_mtom_handler> opcional
2. Elemento <transport> opcional
3. Elemento <service>
4. Elemento <apphandler> opcional
5. Elemento <apphandler_class> opcional
6. O elemento <service_parameter_list> opcional, contendo elementos XML que são disponibilizados para todos os manipuladores de mensagens no enfileiramento no contêiner DFH-SERVICEPLIST.

Exemplo

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

O elemento <provider_pipeline_json>:

Especifica o elemento raiz do documento XML que descreve a configuração do pipeline CICS para um provedor de serviços da web JSON do z/OS Connect.

Isso difere do elemento <provider_pipeline> pelo fato de que os programas do manipulador não podem ser definidos. Esse estilo de pipeline é usado como um contêiner para os recursos **WEBSERVICE** que são usados pelo z/OS Connect. Uma tentativa de iniciar um pipeline <provider_pipeline_json> usando algo diferente de z/OS Connect resultará em um erro. O recurso **PIPELINE** resultante não pode ser usado como o destino de um recurso **USAGE(PIPELINE) URIMAP**. Ele só pode ser usado com recursos **USAGE(JVMSEVER) URIMAP**.

Utilizado em:

- Provedor de Serviços

Atributos:

java_parser={yes|no}

Selecione o tipo de analisador JSON que é usado para processar mensagens de entrada.

Os valores possíveis são:

- yes** Execute a análise JSON usando Java no servidor JVM. Este é o padrão.
- no** Execute análise não Java da mensagem JSON.

Nota: O atributo java_parser é opcional. Se você não fornecê-lo, o comportamento padrão é analisar a mensagem JSON usando Java no servidor JVM. Isso seria o mesmo que especificar java_parser="yes".

java_generator={yes|no}

Selecione o tipo de gerador JSON que é usado para gerar mensagens não enviadas.

Os valores possíveis são:

- yes** Execute a geração JSON usando Java no servidor JVM.
- no** Execute geração não Java da mensagem JSON. Este é o padrão.

Contém:

- Um elemento <jvmserver> contendo o nome do recurso JVMSERVER no qual o z/OS Connect é configurado.

Exemplo que usa a análise Java

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

Exemplo que usa a análise não Java

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

O Elemento <terminal_handler>:

Contém a definição do manipulador de mensagens do terminal do pipeline do provedor de serviços.

Utilizado em:

- Provedor de Serviços

Contido por:

- Elemento <service>

Contém:

Um dos seguintes 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>
```

Se você espera que seu pipeline processe mensagens SOAP 1.1 e SOAP 1.2, você deverá usar o elemento <cics_soap_1.2_handler> ou <cics_soap_1.2_handler_java>.

Lembre-se: Em um provedor de serviços, é possível especificar <cics_soap_1.1_handler> e <cics_soap_1.2_handler> no elemento <service_handler_list>, bem como no elemento <terminal_handler>. No entanto, em um provedor de serviços, é possível especificar somente <cics_soap_1.1_handler_java> e <cics_soap_1.2_handler_java> no elemento <terminal_handler>.

Exemplo

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

Exemplo: ativando o processamento não Java de mensagens JSON

Para ativar o processamento não Java de mensagens JSON, especifique o programa manipulador de terminal como DFHPIJT:

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

Nota: Quando você usar DFHPIJT como o manipulador de terminal, não defina um manipulador de aplicativo no arquivo de configuração de pipeline, ou seja, o arquivo de configuração de pipeline não deve conter um elemento <apphandler>. Se um manipulador de aplicativo for especificado, ele não será chamado.

O Elemento <transport_handler_list>:

Contém uma lista de manipuladores de mensagem que são chamados quando um recurso nomeado é usado.

- Para o transporte MQ, o recurso nomeado é o nome da fila de entrada local.
- Para o transporte HTTP, o recurso é o TCPIPSERVICE que define a porta na qual a solicitação foi recebida.

Utilizado em:

- Provedor de Serviços

Contido por:

- Elemento <named_transport_entry>

Contém:

- Um ou mais elementos <handler>.

Exemplo

```
<transport_handler_list>
  <handler>
    ...
  </handler>
  <handler>
    ...
  </handler>
</transport_handler_list>
```

Elementos Usados em Solicitantes de Serviço

Alguns dos elementos XML usados em um arquivo de configuração de pipeline se aplicam somente aos pipelines do solicitante de serviço.

O elemento <requester_pipeline>:

O elemento-raiz do documento XML que descreve a configuração de um enfileiramento em um solicitante de serviços.

Utilizado em:

- Solicitante de serviços

Contém:

1. Elemento <service> opcional
2. Elemento <transport> opcional
3. Elemento <cics_mtom_handler> opcional
4. O elemento <service_parameter_list> opcional, contendo os elementos XML que são disponibilizados para os manipuladores de mensagens no contêiner DFH-SERVICEPLIST.

Exemplo

```
<requester_pipeline>
  <service>
    <service_handler_list>
      <cics_soap_1.1_handler/>
    </service_handler_list>
  </service>
</requester_pipeline>
```

Elementos Usados em Pipelines do Provedor de Serviços e do Solicitante de Serviço

Alguns dos elementos XML usados em um arquivo de configuração de pipeline se aplicam a ambos os pipelines, do provedor de serviços e do solicitante de serviço.

O Elemento <addressing>:

Especifica o suporte para Web Services Addressing no processamento de SOAP baseado em Java.

Utilizado em:

- Provedor de Serviços
- Solicitante de serviços

Contido por:

Elemento <cics_soap_1.1_handler_java>

Elemento<cics_soap_1.2_handler_java>

Contém:

Um elemento <namespace>. Em um provedor de serviços, este elemento é opcional. O elemento contém um dos dois esquemas WS-Addressing que são suportados pelo CICS. Para mensagens de entrada, Axis2 suporta ambas as especificações. Para mensagens de saída, o namespace especificado neste elemento é usado. Se você não especificar este elemento ou tiver dois elementos, o CICS usará a mesma especificação na mensagem de saída que a mensagem de entrada. Em um solicitante de serviço, este elemento é necessário e você pode especificar somente um namespace para a mensagem de saída.

Este exemplo mostra a configuração para um pipeline do provedor de serviços, no qual ambas as especificações de WS-Addressing são suportadas. O CICS usa a mesma especificação na mensagem de saída que a mensagem de entrada. É possível obter os mesmos resultados especificando um elemento <addressing> vazio.

```
<addressing>
  <namespace>http://www.w3.org/2005/08/addressing</namespace>
  <namespace>http://schemas.xmlsoap.org/ws/2004/08/addressing</namespace>
</addressing>
```


O elemento <cics_json_handler_java>:

Especifica os atributos do programa manipulador para mensagens JSON em pipelines baseados em JSON.

Utilizado em:

- Provedor de Serviços

Contido por:

Contém:

1. Um elemento <jvmserver>.
2. Um elemento <repository> opcional.

Exemplo

O exemplo a seguir mostra o XML para o manipulador JSON baseado em Java e seus elementos aninhados:

```
<?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>
```

O Elemento <cics_soap_1.1_handler>:

Especifica os atributos do programa manipulador para mensagens SOAP 1.1 em pipelines não Java

Utilizado em:

- Solicitante de serviços
- Provedor de Serviços

Contido por:

Elemento <service_handler_list>
Elemento <terminal_handler>

Contém:

Zero, um ou mais elementos <headerprogram>. Cada <headerprogram> contém:

1. Um elemento <program_name>, contendo o nome de um programa de processamento de cabeçalho
2. Um elemento <namespace>, o qual é usado com o elemento <localname> a seguir para determinar qual bloco de cabeçalho em uma mensagem SOAP deve ser processado pelo programa de processamento de cabeçalho. O elemento <namespace> contém o URI (Identificador Uniforme de Recursos) do namespace do bloco de cabeçalho.

- Um elemento `<localname>`, o qual é usado com o elemento `<namespace>` precedente para determinar quais blocos de cabeçalho em uma mensagem SOAP devem ser processados pelo programa de processamento de cabeçalho. O `<localname>` contém o nome de elemento do bloco de cabeçalho.

Por exemplo, considere este bloco de cabeçalho:

```
<t:myheaderblock xmlns:t="http://mynamespace" ...> .... </t:myheaderblock>
```

- O nome do namespace é `http://mynamespace`
- O nome de elemento é `myheaderblock`

Para que um programa do cabeçalho corresponda a este bloco de cabeçalho, codifique os elementos `<namespace>` e `<localname>` conforme a seguir:

```
<namespace>http://mynamespace</namespace>  
<localname>myheaderblock</localname>
```

É possível codificar um asterisco (*) no elemento `<localname>` para indicar que todos os blocos de cabeçalho no namespace cujos nomes iniciam com uma determinada sequência de caracteres devem ser processados. Por exemplo:

```
<namespace>http://mynamespace</namespace>  
<localname>myhead*</localname>
```

Quando você usa o asterisco no elemento `<localname>`, um cabeçalho em uma mensagem pode corresponder a mais de um elemento `<headerprogram>`. Por exemplo, este bloco de cabeçalho

```
<t:myheaderblock xmlns:t="http://mynamespace" ...> .... </myheaderblock>
```

corresponde a todos os elementos `<headerprogram>` a seguir:

```
<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>
```

Quando este é o caso, o programa de cabeçalho que é executado é aquele especificado no elemento `<headerprogram>` no qual o nome de elemento do bloco de cabeçalho é indicado mais precisamente. No exemplo, é HDRPROG3.

Quando a mensagem SOAP contém mais de um cabeçalho, o programa de processamento de cabeçalho é chamado uma vez para cada cabeçalho correspondente, mas a sequência na qual os cabeçalhos são processados é indefinida.

Se você codificar dois ou mais elementos `<headerprogram>` que contêm os mesmos `<namespace>` e `<localname>`, mas que especificam programas de cabeçalho diferentes, somente um dos programas de cabeçalho será executado, mas qual dos programas será executado não é definido.

- Um elemento `<mandatory>`, contendo um valor booleano XML (`true` ou `false`). Como alternativa, é possível codificar os valores como 1 ou 0 respectivamente.

true

Durante o processamento da solicitação de serviço em um pipeline do provedor de serviços, e o processamento de resposta do serviço em um pipeline do solicitante de serviço, o programa de processamento de cabeçalho deve ser chamado pelo menos uma vez, mesmo se nenhum dos cabeçalhos nas mensagens SOAP corresponder aos elementos <namespace> e <localname>:

- Se nenhum dos cabeçalhos corresponder, o programa de processamento de cabeçalho será chamado uma vez.
- Se qualquer um dos cabeçalhos corresponderem, o programa de processamento de cabeçalho será chamado uma vez para cada cabeçalho correspondente.

Durante o processamento de solicitação de serviço em um pipeline do solicitante de serviço, e o processamento de resposta do serviço em um pipeline do provedor de serviços, o programa de processamento de cabeçalho deve ser chamado pelo menos uma vez, embora a mensagem SOAP que o CICS cria não possua cabeçalhos inicialmente. Se você deseja incluir cabeçalhos em sua mensagem, deverá assegurar que pelo menos um programa de processamento de cabeçalho seja chamado, especificando <mandatory>true</mandatory> ou <mandatory>1</mandatory>.

false

O programa de processamento de cabeçalho deve ser chamado somente se um ou mais dos cabeçalhos nas mensagens SOAP corresponder aos elementos <namespace> e <localname>:

- Se nenhum dos cabeçalhos corresponder, o programa de processamento de cabeçalho não será chamado.
- Se qualquer um dos cabeçalhos corresponderem, o programa de processamento de cabeçalho será chamado uma vez para cada cabeçalho correspondente.

Exemplo

```
<cics_soap_1.1_handler>
  <headerprogram>
    <program_name> ... </program_name>
    <namespace>...</namespace>
    <localname>...</localname>
    <mandatory>true</mandatory>
  </headerprogram>
</cics_soap_1.1_handler>
```

O elemento <cics_soap_1.1_handler_java>:

Especifica os atributos do programa manipulador para mensagens SOAP 1.1 em pipelines SOAP baseados em Java.

Utilizado em:

- Solicitante de serviços
- Provedor de Serviços

Contido por:

Elemento <service_handler_list>
Elemento <terminal_handler>

Contém:

1. Um elemento <jvmserver>.
2. Um elemento <repository> opcional.
3. Um elemento <addressing> opcional. Se você ativar o Web Services Addressing em Axis2, não use o programa de processamento de cabeçalho DFHWSADH.
4. Zero, um ou mais elementos <headerprogram>. Cada elemento <headerprogram> contém:
 - a. Um elemento <program_name>, contendo o nome de um programa de processamento de cabeçalho. É possível gravar cabeçalhos Axis2 em Java para processar os cabeçalhos SOAP.
 - b. Um elemento <namespace>, o qual é usado com o elemento <localname> a seguir para determinar qual bloco de cabeçalho em uma mensagem SOAP deve ser processado pelo programa de processamento de cabeçalho. O elemento <namespace> contém o URI (Identificador Uniforme de Recursos) do namespace do bloco de cabeçalho.
 - c. Um elemento <localname>, o qual é usado com o elemento <namespace> precedente para determinar quais blocos de cabeçalho em uma mensagem SOAP devem ser processados pelo programa de processamento de cabeçalho. O <localname> contém o nome de elemento do bloco de cabeçalho.

Por exemplo, considere este bloco de cabeçalho:

```
<t:myheaderblock xmlns:t="http://mynamespace" ...> .... </t:myheaderblock>
```

O nome do namespace é http://mynamespace e o nome de elemento é myheaderblock.

Para que um programa do cabeçalho corresponda a este bloco de cabeçalho, codifique os elementos <namespace> e <localname> conforme a seguir:

```
<namespace>http://mynamespace</namespace>  
<localname>myheaderblock</localname>
```

É possível codificar um asterisco (*) no elemento <localname> para indicar que todos os blocos de cabeçalho no namespace cujos nomes iniciam com uma determinada sequência de caracteres devem ser processados. Por exemplo:

```
<namespace>http://mynamespace</namespace>  
<localname>myhead*</localname>
```

Quando você usa o asterisco no elemento <localname>, um cabeçalho em uma mensagem pode corresponder a mais de um elemento <headerprogram>. Por exemplo, este bloco de cabeçalho:

```
<t:myheaderblock xmlns:t="http://mynamespace" ...> .... </myheaderblock>
```

corresponde a todos os elementos <headerprogram> a seguir:

```
<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>

```

Quando este é o caso, o programa de cabeçalho que é executado é aquele especificado no elemento <headerprogram> no qual o nome de elemento do bloco de cabeçalho é indicado mais precisamente. No exemplo, é HDRPROG3.

Quando a mensagem SOAP contém mais de um cabeçalho, o programa de processamento de cabeçalho é chamado uma vez para cada cabeçalho correspondente, mas a sequência na qual os cabeçalhos são processados é indefinida.

Se você codificar dois ou mais elementos <headerprogram> que contêm os mesmos elementos <namespace> e <localname>, mas que especificam programas de cabeçalho diferentes, somente um dos programas de cabeçalho será executado, mas qual dos programas será executado não é definido.

- d. Um elemento <mandatory>, contendo um valor booleano XML (true ou false). Como alternativa, é possível codificar os valores como 1 ou 0 respectivamente.

true

Durante o processamento da solicitação de serviço em um pipeline do provedor de serviços, e o processamento de resposta do serviço em um pipeline do solicitante de serviço, o programa de processamento de cabeçalho deve ser chamado pelo menos uma vez, mesmo se nenhum dos cabeçalhos nas mensagens SOAP corresponder aos elementos <namespace> e <localname>:

- Se nenhum dos cabeçalhos corresponder, o programa de processamento de cabeçalho será chamado uma vez.
- Se qualquer um dos cabeçalhos corresponderem, o programa de processamento de cabeçalho será chamado uma vez para cada cabeçalho correspondente.

Durante o processamento de solicitação de serviço em um pipeline do solicitante de serviço, e o processamento de resposta do serviço em um pipeline do provedor de serviços, o programa de processamento de cabeçalho deve ser chamado pelo menos uma vez, embora a mensagem SOAP que o CICS cria não possua cabeçalhos inicialmente. Se você desejar incluir cabeçalhos em sua mensagem, deverá assegurar que pelo menos um programa de processamento de cabeçalho seja chamado, especificando <mandatory>true</mandatory> ou <mandatory>1</mandatory>.

false

O programa de processamento de cabeçalho deve ser chamado somente se um ou mais dos cabeçalhos nas mensagens SOAP corresponder aos elementos <namespace> e <localname>:

- Se nenhum dos cabeçalhos corresponder, o programa de processamento de cabeçalho não será chamado.
- Se qualquer um dos cabeçalhos corresponderem, o programa de processamento de cabeçalho será chamado uma vez para cada cabeçalho correspondente.

Exemplo

O exemplo a seguir mostra o XML para o manipulador SOAP baseado em Java e seus elementos aninhados:

```
<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>
```

O Elemento <cics_soap_1.2_handler>:

Especifica os atributos do programa manipulador para mensagens SOAP 1.2 no pipeline não Java.

Utilizado em:

- Solicitante de serviços
- Provedor de Serviços

Contido por:

Elemento <service_handler_list>
Elemento <terminal_handler>

Contém:

Zero, um ou mais elementos <headerprogram>. Cada <headerprogram> contém:

1. Um elemento <program_name>, contendo o nome de um programa de processamento de cabeçalho
2. Um elemento <namespace>, o qual é usado com o elemento <localname> a seguir para determinar qual bloco de cabeçalho em uma mensagem SOAP deve ser processado pelo programa de processamento de cabeçalho. O elemento <namespace> contém o URI (Identificador Uniforme de Recursos) do namespace do bloco de cabeçalho.
3. Um elemento <localname>, o qual é usado com o elemento <namespace> precedente para determinar quais blocos de cabeçalho em uma mensagem SOAP devem ser processados pelo programa de processamento de cabeçalho. O <localname> contém o nome de elemento do bloco de cabeçalho.

Por exemplo, considere este bloco de cabeçalho:

```
<t:myheaderblock xmlns:t="http://mynamespace" ...> .... </t:myheaderblock>
```

- O nome do namespace é http://mynamespace
- O nome de elemento é myheaderblock

Para que um programa do cabeçalho corresponda a este bloco de cabeçalho, codifique os elementos <namespace> e <localname> conforme a seguir:

```
<namespace>http://mynamespace</namespace>
<localname>myheaderblock</localname>
```

É possível codificar um asterisco (*) no elemento <localname> para indicar que todos os blocos de cabeçalho no namespace cujos nomes iniciam com uma determinada sequência de caracteres devem ser processados. Por exemplo:

```
<namespace>http://mynamespace</namespace>
<localname>myhead*</localname>
```

Quando você usa o asterisco no elemento `<localname>`, um cabeçalho em uma mensagem pode corresponder a mais de um elemento `<headerprogram>`. Por exemplo, este bloco de cabeçalho

```
<t:myheaderblock xmlns:t="http://mynamespace" ...> .... </myheaderblock>
```

corresponde a todos os elementos `<headerprogram>` a seguir:

```
<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>
```

Quando este é o caso, o programa de cabeçalho que é executado é aquele especificado no elemento `<headerprogram>` no qual o nome de elemento do bloco de cabeçalho é indicado mais precisamente. No exemplo, é HDRPROG3.

Quando a mensagem SOAP contém mais de um cabeçalho, o programa de processamento de cabeçalho é chamado uma vez para cada cabeçalho correspondente, mas a sequência na qual os cabeçalhos são processados é indefinida.

Se você codificar dois ou mais elementos `<headerprogram>` que contêm os mesmos `<namespace>` e `<localname>`, mas que especificam programas de cabeçalho diferentes, somente um dos programas de cabeçalho será executado, mas qual dos programas será executado não é definido.

4. Um elemento `<mandatory>`, contendo um valor booleano XML (`true` ou `false`). Como alternativa, é possível codificar os valores como 1 ou 0 respectivamente.

true

Durante o processamento da solicitação de serviço em um pipeline do provedor de serviços, e o processamento de resposta do serviço em um pipeline do solicitante de serviço, o programa de processamento de cabeçalho deve ser chamado pelo menos uma vez, mesmo se nenhum dos cabeçalhos nas mensagens SOAP corresponder aos elementos `<namespace>` e `<localname>`:

- Se nenhum dos cabeçalhos corresponder, o programa de processamento de cabeçalho será chamado uma vez.
- Se qualquer um dos cabeçalhos corresponderem, o programa de processamento de cabeçalho será chamado uma vez para cada cabeçalho correspondente.

Durante o processamento de solicitação de serviço em um pipeline do solicitante de serviço, e o processamento de resposta do serviço em um pipeline do provedor de serviços, o programa de processamento de cabeçalho deve ser chamado pelo menos uma vez, embora a mensagem SOAP que o CICS cria não possua cabeçalhos inicialmente. Se você desejar incluir cabeçalhos em sua mensagem, deverá assegurar que pelo menos um

programa de processamento de cabeçalho seja chamado, especificando `<mandatory>true</mandatory>` ou `<mandatory>1</mandatory>`.

false

O programa de processamento de cabeçalho deve ser chamado somente se um ou mais dos cabeçalhos nas mensagens SOAP corresponder aos elementos `<namespace>` e `<localname>`:

- Se nenhum dos cabeçalhos corresponder, o programa de processamento de cabeçalho não será chamado.
- Se qualquer um dos cabeçalhos corresponderem, o programa de processamento de cabeçalho será chamado uma vez para cada cabeçalho correspondente.

Exemplo

```
<cics_soap_1.2_handler>
  <headerprogram>
    <program_name> ... </program_name>
    <namespace>...</namespace>
    <localname>...</localname>
    <mandatory>true</mandatory>
  </headerprogram>
</cics_soap_1.2_handler>
```

O elemento `<cics_soap_1.2_handler_java>`:

Especifica os atributos do programa manipulador para mensagens SOAP 1.2 em pipelines SOAP baseados em Java.

Utilizado em:

- Solicitante de serviços
- Provedor de Serviços

Contido por:

Elemento `<service_handler_list>`

Elemento `<terminal_handler>`

Contém:

1. Um elemento `<jvmserver>`.
2. Um elemento `<repository>` opcional.
3. Um elemento `<addressing>` opcional. Se você ativar o suporte para Web Services Addressing no Axis2, não use programas de processamento de cabeçalho. É possível gravar cabeçalhos Axis2 em Java para processar os cabeçalhos SOAP.
4. Zero, um ou mais elementos `<headerprogram>`. Cada elemento `<headerprogram>` contém:
 - a. Um elemento `<program_name>`, contendo o nome de um programa de processamento de cabeçalho
 - b. Um elemento `<namespace>`, o qual é usado com o elemento `<localname>` a seguir para determinar qual bloco de cabeçalho em uma mensagem SOAP deve ser processado pelo programa de processamento de cabeçalho. O elemento `<namespace>` contém o URI (Identificador Uniforme de Recursos) do namespace do bloco de cabeçalho.
 - c. Um elemento `<localname>`, o qual é usado com o elemento `<namespace>` precedente para determinar quais blocos de cabeçalho em uma mensagem

SOAP devem ser processados pelo programa de processamento de cabeçalho. O `<localname>` contém o nome de elemento do bloco de cabeçalho.

Por exemplo, considere este bloco de cabeçalho:

```
<t:myheaderblock xmlns:t="http://mynamespace" ...> .... </t:myheaderblock>
```

O nome do namespace é `http://mynamespace` e o nome de elemento é `myheaderblock`

Para que um programa do cabeçalho corresponda a este bloco de cabeçalho, codifique os elementos `<namespace>` e `<localname>` conforme a seguir:

```
<namespace>http://mynamespace</namespace>  
<localname>myheaderblock</localname>
```

É possível codificar um asterisco (*) no elemento `<localname>` para indicar que todos os blocos de cabeçalho no namespace cujos nomes iniciam com uma determinada sequência de caracteres devem ser processados. Por exemplo:

```
<namespace>http://mynamespace</namespace>  
<localname>myhead*</localname>
```

Quando você usa o asterisco no elemento `<localname>`, um cabeçalho em uma mensagem pode corresponder a mais de um elemento `<headerprogram>`. Por exemplo, este bloco de cabeçalho:

```
<t:myheaderblock xmlns:t="http://mynamespace" ...> .... </myheaderblock>
```

corresponde a todos os elementos `<headerprogram>` a seguir:

```
<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>
```

Quando este é o caso, o programa de cabeçalho que é executado é aquele especificado no elemento `<headerprogram>` no qual o nome de elemento do bloco de cabeçalho é indicado mais precisamente. No exemplo, é HDRPROG3.

Quando a mensagem SOAP contém mais de um cabeçalho, o programa de processamento de cabeçalho é chamado uma vez para cada cabeçalho correspondente, mas a sequência na qual os cabeçalhos são processados é indefinida.

Se você codificar dois ou mais elementos `<headerprogram>` que contêm os mesmos elementos `<namespace>` e `<localname>`, mas que especificam programas de cabeçalho diferentes, somente um dos programas de cabeçalho será executado, mas qual dos programas será executado não é definido.

- d. Um elemento `<mandatory>`, contendo um valor booleano XML (`true` ou `false`). Como alternativa, é possível codificar os valores como 1 ou 0 respectivamente.

true

Durante o processamento da solicitação de serviço em um pipeline do provedor de serviços, e o processamento de resposta do serviço em um pipeline do solicitante de serviço, o programa de processamento de cabeçalho deve ser chamado pelo menos uma vez, mesmo se nenhum dos cabeçalhos nas mensagens SOAP corresponder aos elementos `<namespace>` e `<localname>`:

- Se nenhum dos cabeçalhos corresponder, o programa de processamento de cabeçalho será chamado uma vez.
- Se qualquer um dos cabeçalhos corresponderem, o programa de processamento de cabeçalho será chamado uma vez para cada cabeçalho correspondente.

Durante o processamento de solicitação de serviço em um pipeline do solicitante de serviço, e o processamento de resposta do serviço em um pipeline do provedor de serviços, o programa de processamento de cabeçalho deve ser chamado pelo menos uma vez, embora a mensagem SOAP que o CICS cria não possua cabeçalhos inicialmente. Se você deseja incluir cabeçalhos em sua mensagem, deverá assegurar que pelo menos um programa de processamento de cabeçalho seja chamado, especificando `<mandatory>true</mandatory>` ou `<mandatory>1</mandatory>`.

false

O programa de processamento de cabeçalho deve ser chamado somente se um ou mais dos cabeçalhos nas mensagens SOAP corresponder aos elementos `<namespace>` e `<localname>`:

- Se nenhum dos cabeçalhos corresponder, o programa de processamento de cabeçalho não será chamado.
- Se qualquer um dos cabeçalhos corresponderem, o programa de processamento de cabeçalho será chamado uma vez para cada cabeçalho correspondente.

Exemplo

O exemplo a seguir mostra o XML para o manipulador SOAP baseado em Java e seus elementos aninhados:

```
<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>
```

O Elemento `<default_http_transport_handler_list>`:

Especifica os manipuladores de mensagem que são chamados por padrão quando o transporte HTTP está em uso.

Em um provedor de serviços, os manipuladores de mensagem especificados nesta lista são chamados somente se a lista de manipuladores definidos em um elemento `<named_transport_entry>` é menos específica.

Utilizado em:

- Provedor de Serviços
- Solicitante de serviços

Contido por:

- Elemento `<transport>`

Contém:

- Um ou mais elementos `<handler>`.

Exemplo

```
<default_http_transport_handler_list>
  <handler>
    ...
  </handler>
  <handler>
    ...
  </handler>
</default_http_transport_handler_list>
```

O Elemento `<default_mq_transport_handler_list>`:

Especifica os manipuladores de mensagem que são chamados por padrão quando o transporte do WebSphere MQ está em uso.

Em um provedor de serviços, os manipuladores de mensagem especificados nesta lista são chamados somente se a lista de manipuladores definidos em um elemento `<named_transport_entry>` é menos específica.

Utilizado em:

- Provedor de Serviços
- Solicitante de serviços

Contido por:

- Elemento `<transport>`

Contém:

- Um ou mais elementos `<handler>`.

Exemplo

```
<default_mq_transport_handler_list>
  <handler>
    ...
  </handler>
  <handler>
    ...
  </handler>
</default_mq_transport_handler_list>
```

O Elemento `<default_transport_handler_list>`:

Especifica os manipuladores de mensagens que são chamados por padrão quando qualquer transporte está em uso.

Em um provedor de serviços, manipuladores de mensagens especificados nesta lista são chamados quando a lista de manipuladores definidos em qualquer um dos elementos a seguir são menos específicos:

```
<default_http_transport_handler_list>
<default_mq_transport_handler_list>
<named_transport_entry>
```

Utilizado em:

- Provedor de Serviços
- Solicitante de serviços

Contido por:

- Elemento `<transport>`

Contém:

- Um ou mais elementos `<handler>`.

Exemplo

```
<default_transport_handler_list>
  <handler>
    <program>HANDLER1</program>
    <handler_parameter_list/>
  </handler>
  <handler>
    <program>HANDLER2</program>
    <handler_parameter_list/>
  </handler>
</default_transport_handler_list>
```

O Elemento `<handler>`:

Especifica os atributos de um programa do manipulador de mensagem.

Alguns programas de manipulador fornecidos pelo CICS não usam o elemento `<handler>`. Por exemplo, os programas do manipulador de mensagem SOAP fornecidos pelo CICS são definidos usando os elementos `<cics_soap_1.1_handler>`, `<cics_soap_1.2_handler>`, `<cics_soap_1.1_handler_java>` e `<cics_soap_1.2_handler_java>`.

Utilizado em:

- Provedor de Serviços
- Solicitante de serviços

Contido por:

```
<default_transport_handler_list>
<transport_handler_list>
<service_handler_list>
<terminal_handler>
<default_http_transport_handler_list>
```

<default_mq_transport_handler_list>

Contém:

1. Elemento <program>, contendo o nome do programa do manipulador
2. Elemento <handler_parameter_list>, contendo elementos XML que se tornam disponíveis para os manipuladores de mensagem no contêiner DFH-HANDLERPLIST.

Exemplo

```
<?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>
```

Neste exemplo, o programa manipulador é MYPROG. A lista de parâmetros do manipulador consiste em um único elemento <output>; o conteúdo da lista de parâmetros é conhecido como MYPROG.

O elemento <jvmserver>:

Especifica o nome do recurso JVMSERVER.

Este elemento identifica o nome do recurso JVMSERVER, que processará a solicitação. Se um valor não for fornecido, uma mensagem de erro será gerada e o PIPELINE será instalado no estado DISABLED.

Utilizado em:

- Provedor de Serviços
- Solicitante de serviços

Contido por:

- Elemento do O elemento <cics_json_handler_java>
- Elemento do O elemento <cics_soap_1.1_handler_java>
- Elemento do O elemento <cics_soap_1.2_handler_java>
- Elemento do

Exemplo

```
<jvmserver>JVMSERVER_NAME</jvmserver>
```

O Elemento <repository>:

Especifica o nome de diretório do repositório Axis2.

Este elemento opcional identifica o nome de diretório do repositório Axis2. Se usar esta opção, você deverá especificar O elemento `<jvmserver>` antecipadamente no manipulador XML. Se o elemento não for fornecido, o repositório de amostra será usado. Ao instalar o CICS Transaction Server, o repositório Axis2 de amostra é instalado no diretório `/usr/lpp/cicsts/cicsts54/lib/pipeline/repository`, em que `/usr/lpp/cicsts/cicsts54` é o diretório de instalação padrão para arquivos CICS no z/OS UNIX.

Utilizado em:

- Provedor de Serviços
- Solicitante de serviços

Contido por:

- O elemento `<cics_json_handler_java>`
- O elemento `<cics_soap_1.1_handler_java>`
- O elemento `<cics_soap_1.2_handler_java>`

Exemplo

```
<cics_soap_1.1_handler_java>
  <jvmserver>JVMSERV1</jvmserver>
  <repository>/lib/pipeline/repository</repository>
</cics_soap_1.1_handler_java>
```

O Elemento `<service>`:

Especifica os manipuladores de mensagem que são chamados para cada solicitação.

Utilizado em:

- Provedor de Serviços
- Solicitante de serviços

Contido por:

```
  <provider_pipeline>
  <requester_pipeline>
```

Contém:

1. Elemento `<service_handler_list>`
2. Apenas em um provedor de serviços, um elemento `<terminal_handler>`

Exemplo

```
<service>
  <service_handler_list>
  ...
</service_handler_list>
  <terminal_handler>
  ...
</terminal_handler>
</service>
```

O Elemento `<service_handler_list>`:

Especifica uma lista de manipuladores de mensagem que são chamados para cada solicitação.

Utilizado em:

- Provedor de Serviços
- Solicitante de serviços

Contido por:

- Elemento <service>

Contém:

Um ou mais dos elementos a seguir:

```
<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>
```

Você determina a ordem em que cada manipulador é chamado no tempo de execução pela ordem em que especifica os elementos do manipulador no elemento <service_handler_list>. Por exemplo, se seu pipeline suportar WS-Security, mensagens SOAP criptografadas permanecerão criptografadas até o elemento <wsse_handler> ser chamado. Portanto, você deve especificar o elemento <wsse_handler> antes de qualquer outro programa manipulador que processe mensagens não criptografadas.

O elemento <service_handler_list> para um provedor de serviços não pode conter os elementos <cics_soap_1.1_handler_java> e <cics_soap_1.2_handler_java>, porque esses elementos devem ser especificados no elemento <terminal_handler> para pipelines baseados em Java. Um solicitante de serviço pode conter <cics_soap_1.1_handler_java> e <cics_soap_1.2_handler_java>, no entanto, se estes elementos forem usados, eles deverão ser o primeiro elemento listado no elemento <service_handler_list>.

Se você espera que seu pipeline processe mensagens SOAP 1.1 e SOAP 1.2, você deverá usar o elemento <cics_soap_1.2_handler> ou <cics_soap_1.2_handler_java>.

É possível usar um manipulador SOAP 1.1 ou SOAP 1.2 em um pipeline do solicitante de serviço, mas neste caso o manipulador SOAP 1.2 não suporta mensagens SOAP 1.1. Não especifique o manipulador SOAP 1.1 ou SOAP 1.2 no pipeline se seus aplicativos do solicitante de serviço estiverem enviando envelopes SOAP completos no contêiner DFHREQUEST. Isto evita duplicar os cabeçalhos de mensagem SOAP em mensagens de saída.

Em um provedor de serviços, é possível especificar o manipulador genérico e manipuladores SOAP no elemento <terminal_handler> bem como no elemento <service_handler_list>. Para obter mais informações sobre como processar o cabeçalho SOAP, consulte “Programas de Processamento de Cabeçalho” na página 143.

Exemplo

```
<service_handler_list>  
  <wsse_handler>  
  ...
```

```

    </wsse_handler>
    <cics_soap_1.1_handler_java>
        ...
    </cics_soap_1.1_handler_java>
    <handler>
        ...
    </handler>
</service_handler_list>

```

O Elemento <service_parameter_list>:

Especifica os elementos XML que são disponibilizados para todos os manipuladores de mensagem no pipeline no contêiner DFH-SERVICEPLIST. Este é um elemento opcional.

Utilizado em:

- Solicitante de serviços
- Provedor de Serviços

Contém:

- Se estiver usando WS-AT: um elemento <registration_service_endpoint>
- Em um solicitante de serviço se estiver usando WS-AT: um elemento <new_tx_context_required/> opcional
- Tags definidas pelo usuário opcionais

Exemplo

```

<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>

```

O Elemento <transport>:

Especifica manipuladores que devem ser chamados somente quando um transporte específico está em uso.

Utilizado em:

- Provedor de Serviços
- Solicitante de serviços

Contido por:

```

  <provider_pipeline>
  <requester_pipeline>

```

Contém:

Em um provedor de serviços:

1. Um elemento <default_transport_handler_list> opcional
2. Um elemento <default_http_transport_handler_list> opcional

3. Um elemento `<default_mq_transport_handler_list>` opcional
4. Zero, um ou mais elementos `<named_transport_entry>`

Em um solicitante de serviço:

1. Um elemento `<default_target>` opcional. O `<default_target>` contém um URI que o CICS usa para localizar o serviço da web de destino quando o aplicativo do solicitante de serviço não fornece um URI. Em muitos casos, no entanto, o URI do destino será fornecido pelo aplicativo do solicitante de serviço e o que você especificar no `<default_target>` será ignorado. Por exemplo, os aplicativos do provedor de serviços que são implementados usando o assistente de serviços da web do CICS geralmente obtêm o URI da descrição de serviços da web.
2. Um elemento `<default_http_transport_handler_list>` opcional
3. Um elemento `<default_mq_transport_handler_list>` opcional
4. Um elemento `<default_transport_handler_list>` opcional

Exemplo

```
<transport>
  <default_transport_handler_list>
    ...
  </default_transport_handler_list>
</transport>
```

Configuração de Pipeline para MTOM/XOP

Os pipelines SOAP do CICS podem suportar as especificações Message Transmission Optimization Mechanism (MTOM) e XML-binary Optimized Packaging (XOP). Estas especificações definem um mecanismo para enviar e receber dados binários usando SOAP, sem incorrer na sobrecarga da codificação base64. Para ativar o suporte a MTOM, você deve configurar seus pipelines de acordo.

O Elemento `<mtom>`:

Ativa o suporte de MTOM/XOP para pipelines baseados em Java. Se este elemento for definido no arquivo de configuração de pipeline, o suporte de MTOM será ativado para todas as mensagens de entrada e saída. Entretanto, se este elemento não for especificado no arquivo de configuração de pipeline, o suporte de MTOM será ativado somente para mensagens de entrada.

Utilizado em:

- Provedor de Serviços
- Solicitante de serviços

Contido por:

```
<cics_soap_1.1_handler_java>
<cics_soap_1.2_handler_java>
```

Para ambos os arquivos de configuração de pipeline do provedor e do solicitante, o elemento `<mtom>` deve ser definido após o elemento `<addressing>` opcional e antes do elemento `<headerprogram>` opcional.

Exemplo

Para um pipeline do modo de provedor ou solicitante, você poderá 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>

```

O Elemento <cics_mtom_handler>:

Ativa o programa manipulador MTOM fornecido para pipelines SOAP. Este programa fornece suporte para mensagens multiparte/relacionadas MIME MTOM que contêm documentos XOP e anexos binários. O suporte a MTOM é ativado para todas as mensagens de entrada recebidas no enfileiramento, mas o suporte a MTOM para mensagens de saída é condicionalmente ativado e está sujeito a outras opções.

Utilizado em:

- Provedor de Serviços
- Solicitante de serviços

Contido por:

```

  <provider_pipeline>
  <requester_pipeline>

```

Em um arquivo de configuração do enfileiramento do provedor, o elemento <cics_mtom_handler> deve ser definido antes do elemento <transport>. No tempo de execução, o programa do manipulador do MTOM precisa desempacotar a mensagem do MTOM antes de outros manipuladores processá-la, incluindo o manipulador de transporte. Ele é, então, chamado como o último manipulador para a mensagem de resposta, para empacotar uma mensagem MTOM para enviar ao solicitante de serviço da web.

Em um arquivo de configuração de pipeline do solicitante, o elemento <cics_mtom_handler> deve ser definido após o elemento <transport>. No tempo de execução, a mensagem de pedido de saída não é convertida no formato MTOM até que todos os outros manipuladores a tenham processado. Ele é, então, chamado como o primeiro manipulador para a mensagem de resposta de entrada para descompactar a mensagem MTOM antes que outros manipuladores a processem e retornem ao programa solicitante.

Nota: Você não deve usar este programa manipulador com pipelines baseados em Java. Para pipelines baseados em Java, especifique o elemento <mtom>.

Contém:

```

  elemento <dfhmtom_configuration>

```

As opções padrão podem ser alteradas utilizando as opções de configuração especificadas no elemento <dfhmtom_configuration>. Se não desejar alterar as opções padrão, você poderá utilizar um elemento vazio.

Exemplo

Para um enfileiramento do modo do provedor, você poderá especificar:

```
<provider_pipeline>
  <cics_mtom_handler></cics_mtom_handler>
  <transport>
    ....
  </transport>
  <service>
    ....
  </service>
</provider_pipeline>
```

O elemento <dfhmtom_configuration>:

Especifica informações de configuração para o programa manipulador do MTOM fornecido para pipelines que não suportam Java. Este programa fornece suporte para mensagens MIME que contêm documentos XOP e anexos binários. Se não especificar nenhuma configuração para o MTOM, o CICS assumirá valores padrão.

Utilizado em:

- Provedor de Serviços
- Solicitante de serviços

Contido por:

<cics_mtom_handler>

Atributos:

Nome	Descrição
versão	Um número inteiro que detona a versão das informações de configuração. O único valor válido é 1.

Contém:

- Um elemento <mtom_options> opcional
- Um elemento <xop_options> opcional
- Um elemento <mime_options> opcional

Exemplo

```
<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>
```

O Elemento <mtom_options>:

Especifica quando usar MTOM para mensagens SOAP de saída para pipelines que não suportam Java.

Utilizado em:

- Provedor de Serviços
- Solicitante de serviços

Contido por:

<dfhmtom_configuration>

Atributos:

Atributo	Descrição
send_mtom	<p>Especifica se o MTOM deverá ser utilizado para converter a mensagem SOAP de saída em uma mensagem MIME:</p> <p>não O MTOM não é utilizado para mensagens SOAP de saída.</p> <p>same No modo provedor de serviços, o MTOM é utilizado para mensagens de respostas SOAP sempre que o solicitante utiliza o MTOM. Esse é valor padrão em um enfileiramento do provedor de serviços.</p> <p>No modo do solicitante de serviço, a especificação deste valor é a mesma que quando você especifica send_mtom="yes".</p> <p>sim O MTOM é utilizado para todas as mensagens SOAP de saída. Esse é o valor padrão em um enfileiramento do solicitante de serviços.</p>
send_when_no_xop	<p>Especifique se uma mensagem do MTOM deve ser enviada, mesmo quando não há anexos binários presentes na mensagem.</p> <p>não O MTOM só é utilizado quando anexos binários estão sendo enviados com a mensagem.</p> <p>sim O MTOM é utilizado para todas as mensagens SOAP de saída, mesmo quando não há nenhum anexo binário a ser enviado na mensagem. Esse é o valor padrão e é basicamente utilizado como um indicador para o programa receptor que o emissor suporta o MTOM/XOP.</p> <p>Este atributo pode ser combinado com qualquer um dos valores de atributo send_mtom, mas não possui efeito se você especificar send_mtom="no".</p>

Exemplo

```
<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>
```

Neste exemplo de pipeline do provedor, as mensagens SOAP são convertidas em mensagens MTOM somente quando anexos binários precisam ser enviados com a mensagem e o solicitante de serviço enviou uma mensagem MTOM.

O Elemento <xop_options>:

Especifica se o processamento de XOP pode ocorrer no modo direto ou de compatibilidade para pipelines que não suportam Java.

Utilizado em:

- Provedor de Serviços
- Solicitante de serviços

Contido por:

<dfhmtom_configuration>

Atributos:

Atributo	Descrição
apphandler_supports_xop	<p>No modo provedor, especifica se o manipulador do aplicativo é capaz de manipular os documentos XOP no modo direto:</p> <p>não O manipulador do aplicativo não pode manipular os documentos XOP diretamente. Esse será o valor padrão se o elemento <apphandler> não especificar o DFHPITP.</p> <p>O modo de compatibilidade é utilizado no enfileiramento para manipular mensagens de entrada ou saída recebidas ou enviadas no formato MTOM.</p> <p>sim O manipulador do aplicativo pode manipular documentos XOP. Esse será o valor padrão se o elemento <apphandler> especificar o DFHPITP.</p> <p>O modo direto é utilizado no enfileiramento para manipular mensagens de entrada e de saída que são recebidas ou enviadas no formato MTOM. Isso estará sujeito a restrições no tempo de execução. Por exemplo, se especificou os elementos relacionados ao WS-Security no arquivo de configuração do enfileiramento, o manipulador MTOM determinará se o enfileiramento deverá utilizar o modo de compatibilidade no lugar do modo direto para processamento de documentos XOP.</p> <p>No modo do solicitante, especifica se aplicativos do solicitante de serviço usam o suporte de serviços da web do CICS para criar e manipular documentos XOP no modo direto.</p> <p>não Os aplicativos do solicitante de serviço não usam o suporte de serviços da web do CICS. Especifique esse valor se o aplicativo do solicitante vincular-se ao DFHPIRT para acionar o enfileiramento e não for, portanto, capaz de criar e manipular documentos XOP no modo direto.</p> <p>sim Os aplicativos do solicitante de serviço usam o suporte de serviços da web do CICS. Especifique este valor se o aplicativo do solicitante utilizar o comando EXEC CICS INVOKE WEBSERVICE.</p>

Exemplo

```
<provider_pipeline>
  <cics_mtom_handler>
    <dfhmtom_configuration version="1">
      <xop_options apphandler_supports_xop="no"/>
```

```

    </dfhmtom_configuration>
  </cics_mtom_handler>
  ...
</provider_pipeline>

```

Nesse exemplo de enfileiramento do provedor, as mensagens do MTOM de entrada e as mensagens de resposta de saída são processadas no enfileiramento utilizando o modo de compatibilidade.

O Elemento `<mime_options>`:

Especifica o nome de domínio que deve ser usado ao gerar valores de ID de conteúdo MIME para pipelines que não suportam Java. Os valores de ID de conteúdo MIME são usados para identificar anexos binários.

Utilizado em:

- Provedor de Serviços
- Solicitante de serviços

Contido por:

```
<dfhmtom_configuration>
```

Atributos:

Atributo	Descrição
content_id_domain	<p>A sintaxe a ser utilizada é <i>domain.name</i>.</p> <p>Para estar em conformidade com padrões da Internet, o nome deve ser um nome de host de Internet válido e ser exclusivo para o sistema CICS no qual o enfileiramento está instalado. Observe que isso não é verificado pelo CICS.</p> <p>Se esse elemento for omitido, o CICS utiliza o valor <code>cicsts</code>.</p>

Exemplo

```

<provider_pipeline>
  <dfhmtom_configuration version="1">
    <mime_options content_id_domain="example.org"/>
  </dfhmtom_configuration>
  ...
</provider_pipeline>

```

Nesse exemplo, as referências a anexos binários são criadas utilizando `cid:unique_value@example.org`.

Configuração de Pipeline para WS-Security

Para que os aplicativos de solicitante e provedor de serviço da web participem dos protocolos WS-Security, você deve configurar seus pipelines adequadamente, incluindo o manipulador de mensagem DFHWSSE, e fornecendo informações de configuração para o manipulador.

Exemplo

Um arquivo de configuração de pipeline do provedor que usa WS-Security pode ter o formato a seguir:

```

<?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>
      <handler>
        ...
      </handler>
    </service_handler_list>
    <terminal_handler>
      <cics_soap_1.2_handler/>
    </terminal_handler>
  </service>
  <apphandler>DFHPITP</apphandler>
</provider_pipeline>

```

O Elemento **<wsse_handler>**:

Especifica parâmetros usados pelo manipulador de mensagem fornecido pelo CICS que fornece suporte para WS-Security.

Utilizado em:

- Provedor de Serviços
- Solicitante de serviços

Contido por:

`<service_handler_list>`

Contém:

- Um elemento `<dfhwsse_configuration>`.

Em um arquivo de configuração de pipeline do provedor, o manipulador de mensagem fornecido pelo CICS para WS-Security pode precisar decriptografar uma mensagem criptografada. O elemento `<wsse_handler>` deve ser definido antes de qualquer outro programa manipulador que precisa processar o conteúdo da mensagem não criptografada.

Em um arquivo de configuração de pipeline do solicitante, o manipulador de mensagem fornecido CICS para WS-Security pode precisar criptografar uma mensagem. Ele deve ser definido após qualquer outro programa manipulador que precise processar o conteúdo da mensagem não criptografado, incluindo o programa manipulador CICS SOAP.

O elemento **<dfhwsse_configuration>**:

Especifica informações de configuração para o manipulador de segurança DFHWSSE1, que fornece suporte para proteger serviços da web.

Utilizado em:

- Provedor de Serviços
- Solicitante de serviços

Contido por:

<wsse_handler>

Atributos:

Nome	Descrição
versão	Um número inteiro que detona a versão das informações de configuração. O único valor válido é 1.

Contém:

1. Um dos seguintes elementos:
 - Um elemento <authentication> opcional.
 - Em um pipeline do solicitante de serviço, o elemento <authentication> especifica o tipo de autenticação que deve ser usado no cabeçalho de segurança de mensagens SOAP de saída.
 - Em um pipeline do provedor de serviços, o elemento especificar se o CICS usa os tokens de segurança em uma mensagem SOAP de entrada para determinar o ID do usuário sob o qual o trabalho é processado.
 - Um elemento <sts_authentication> opcional.

O atributo action neste elemento especifica qual tipo de solicitação enviar para o Serviço do Token de Segurança. Se o pedido estiver para emitir um token de identidade, então o CICS utiliza os valores nos elementos aninhados para solicitar um token de identidade do tipo especificado.
2. Se especificar um elemento <sts_authentication>, também será necessário especificar um elemento <sts_endpoint>.

Quando esse elemento estiver presente, o CICS utilizará o URI no elemento <endpoint> para enviar um pedido ao Security Token Service.
3. Um elemento <expect_signed_body/> opcional e vazio.

O elemento <expect_signed_body/> indica que o <body> da mensagem de entrada deve ser assinado. Se o corpo de uma mensagem de entrada não estiver corretamente assinado, o CICS rejeitará a mensagem com uma falha de segurança.
4. Um elemento <expect_encrypted_body/> opcional e vazio.

O elemento <expect_encrypted_body/> indica que o <body> da mensagem de entrada deve ser criptografado. Se o corpo de uma mensagem de entrada não estiver corretamente criptografado, o CICS rejeitará a mensagem com uma falha de segurança.
5. Um elemento <sign_body> opcional.

Se esse elemento estiver presente, o CICS assinará o <body> da mensagem de saída, utilizando o algoritmo especificado no elemento <algorithm> contido no elemento <sign_body>.
6. Um elemento <encrypt_body> opcional.

Se esse elemento estiver presente, o CICS criptografará o <body> da mensagem de saída, utilizando o algoritmo especificado no elemento <algorithm> contido no elemento <encrypt_body>.
7. Em processos penas de provedor, há um elemento opcional <reject_signature/>.

Se esse elemento estiver presente, o CICS rejeita qualquer mensagem que inclua um certificado em seu cabeçalho que assine parte de ou todo o corpo da mensagem. Uma falha de SOAP é emitida para o solicitante de serviço da web.

8. Em processos somente de provedor, há um elemento opcional `<reject_encryption/>`.

Se esse elemento estiver presente, o CICS rejeita qualquer mensagem que esteja parcial ou completamente criptografada. Uma falha de SOAP é emitida para o solicitante de serviço da web.

Exemplo

```
<dfhwsse_configuration version="1">
  <sts_authentication action="issue">
    <auth_token_type>
      <namespace>http://example.org.tokens</namespace>
      <element>UsernameToken</element>
    </auth_token_type>
    <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>
</dfhwsse_configuration>
```

O elemento <autenticação>:

Especifica a utilização de tokens de segurança nos cabeçalhos de mensagens SOAP de entrada e saída.

Utilizado em:

- Provedor de Serviços
- Solicitante de serviços

Contido por:

`<dfhwsse_configuration>`

Atributos:

Atributo	Descrição
confiança e modo	<p>Juntos, os atributos trust e mode especificam:</p> <ul style="list-style-type: none"> • se a identidade asserida será utilizada • a combinação de tokens de segurança que são utilizados em mensagens SOAP. <p>A identidade declarada permite que um usuário confiável declare que o trabalho deve ser executado sob uma identidade diferente, a <i>identidade declarada</i>, sem que o usuário confiável tenha as credenciais que estão associadas a essa identidade.</p> <p>Quando a identidade asserida é utilizada, as mensagens contêm um <i>token de confiança</i> e um <i>token de identidade</i>. O token de confiança é usado para verificar se o emissor possui as permissões corretas para declarar identidades. O token de identidade contém a identidade declarada, ou seja, o ID do usuário sob o qual a solicitação é executada.</p> <p>A utilização de identidade asserida requer que um provedor de serviços confie no solicitante para fazer esta asserção. No CICS, o relacionamento de confiança é estabelecido com definições substitutas do gerenciador de segurança: a identidade solicitante deve ter a autoridade correta para iniciar o trabalho em nome da identidade asserida.</p> <p>As combinações permitidas desses atributos e seus significados são descritos em Tabela 3 e Tabela 4 na página 123.</p>

Tabela 3. Os Atributos **mode** e **trust** em um Enfileiramento do Solicitante de Serviços

trust	Modo de Execução	Significado
nenhum	nenhum	Nenhuma credencial é incluída na mensagem
nenhum	básicas	<i>Combinação inválida de valores de atributo</i>
nenhum	assinatura	A identidade asserida não é utilizada. O CICS usa um único token de segurança X.509, que é incluído na mensagem e usado para assinar o corpo da mensagem. O certificado é identificado com o elemento <certificate_label> e o algoritmo é especificado no elemento <algorithm>.
blind	nenhum	<i>Combinação inválida de valores de atributo</i>
blind	básicas	A identidade asserida não é utilizada. O CICS inclui um token de identidade na mensagem, mas não fornece um token de confiança. O token de identidade é um nome de usuário sem senha. O ID do usuário colocado no token de identidade é o conteúdo do contêiner DFHWS-USERID (que, por padrão, contém o ID do usuário da tarefa em execução).
blind	assinatura	<i>Combinação inválida de valores de atributo</i>
básicas	nenhum	<i>Combinação inválida de valores de atributo</i>
básicas	básicas	<i>Combinação inválida de valores de atributo</i>
básicas	assinatura	<i>Combinação inválida de valores de atributo</i>
assinatura	nenhum	<i>Combinação inválida de valores de atributo</i>

Tabela 3. Os Atributos **mode** e **trust** em um Enfileiramento do Solicitante de Serviços (continuação)

trust	Modo de Execução	Significado
assinatura	básicas	<p>A identidade asserida é utilizada. O CICS inclui os seguintes tokens na mensagem:</p> <ul style="list-style-type: none"> • O token de confiança é um token de segurança X.509. • O token de identidade é um nome de usuário sem senha. <p>O certificado que é usado para assinar o token de identidade e o corpo da mensagem é especificado pelo <certificate_label>. O ID do usuário colocado no token de identidade é o conteúdo do contêiner DFHWS-USERID (que, por padrão, contém o ID do usuário da tarefa em execução).</p>
assinatura	assinatura	Combinação inválida de valores de atributo

Tabela 4. Os Atributos **mode** e **trust** em um Enfileiramento do Provedor de Serviços

trust	Modo de Execução	Significado
nenhum	nenhum	As mensagens de entrada não precisam conter credenciais e o CICS não tenta extrair ou verificar nenhuma das credenciais que estão localizadas em uma mensagem. No entanto, o CICS verifica se quaisquer elementos assinados foram assinados corretamente.
nenhum	básicas	As mensagens de entrada devem conter um token de segurança de nome do usuário com uma senha. O CICS coloca o nome do usuário no contêiner DFHWS-USERID.
nenhum	ICRX básico	Combinação inválida de valores de atributo
nenhum	basic-kerberos	Combinação inválida de valores de atributo
nenhum	assinatura	As mensagens de entrada devem conter um token de segurança X.509 que tenha sido utilizado para assinar o corpo da mensagem.
blind	nenhum	Combinação inválida de valores de atributo
blind	básicas	As mensagens de entrada devem conter um token de identidade, em que o token de identidade contém um ID do usuário e, opcionalmente, uma senha. O CICS coloca o ID do usuário no contêiner DFHWS-USERID. Se nenhuma senha for incluída, o CICS usará o ID do usuário sem verificá-lo. Se uma senha estiver incluída, o DFHWSSE1 do manipulador de segurança a verificará.
blind	ICRX básico	As mensagens de entrada devem conter um token de identidade ICRX. O CICS resolve a identidade, coloca o ID do usuário no contêiner DFHWS-USERID e coloca o ICRX no contêiner DFHWS-ICRX. A autenticação, se necessária, usa SSL certificado pelo cliente ou um outro protocolo de segurança.
blind	basic-kerberos	Combinação inválida de valores de atributo

Tabela 4. Os Atributos **mode** e **trust** em um Enfileiramento do Provedor de Serviços (continuação)

trust	Modo de Execução	Significado
blind	assinatura	As mensagens de entrada devem conter um token de identidade, em que o token de identidade é o primeiro certificado X.509 no cabeçalho da mensagem SOAP. O certificado não precisa ter assinado a mensagem. O manipulador de segurança extrai o ID do usuário correspondente e o coloca no contêiner DFHWS-USERID.
básicas	nenhum	<i>Combinação inválida de valores de atributo</i>
básicas	básicas	As mensagens de entrada devem utilizar a identidade asserida: <ul style="list-style-type: none"> • O token de confiança é um token do nome de usuário com uma senha • O token de identidade é um segundo token do nome de usuário sem uma senha. O CICS coloca esse nome do usuário no contêiner DFHWS-USERID.
básicas	ICRX básico	As mensagens de entrada devem utilizar a identidade asserida: <ul style="list-style-type: none"> • O token de confiança é um token do nome de usuário com uma senha. <p>O CICS estabelece se a combinação de ID do usuário e senha é válida e, se eles forem válidos, o CICS resolve a identidade baseada em ICRX declarada para um ID do usuário. O CICS, então, executa uma verificação de segurança substituta a partir da identidade autenticada na identidade declarada.</p> <ul style="list-style-type: none"> • O token de identidade é um ICRX, que identifica o usuário cliente específico. O CICS coloca o nome do usuário no contêiner DFHWS-USERID e o ICRX no contêiner DFHWS-ICRX.
básicas	basic-kerberos	As mensagens de entrada devem usar identidade declarada. <p>Um token é necessário, um token Kerberos Versão 5 com um dos tipos de formato a seguir:</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>O token deve ser codificado por Base-64. O CICS valida o token usando o Network Authentication Service para z/OS e coloca o ID do usuário associado ao token no contêiner DFHWS-USERID.</p>

Tabela 4. Os Atributos **mode** e **trust** em um Enfileiramento do Provedor de Serviços (continuação)

trust	Modo de Execução	Significado
básicas	assinatura	As mensagens de entrada devem utilizar a identidade asserida: <ul style="list-style-type: none"> • O token de confiança é um token do nome de usuário com uma senha • O token de identidade é um certificado X.509. O CICS coloca o ID do usuário associado ao certificado no contêiner DFHWS-USERID.
assinatura	nenhum	<i>Combinação inválida de valores de atributo</i>
assinatura	básicas	As mensagens de entrada devem utilizar a identidade asserida: <ul style="list-style-type: none"> • O token de confiança é um certificado X.509 • O token de identidade é um token do nome de usuário sem uma senha. O CICS coloca o nome do usuário no contêiner DFHWS-USERID. <p>O token de identidade e o corpo devem ser assinados com o certificado X.509.</p>
assinatura	ICRX básico	As mensagens de entrada devem usar identidade declarada. <ul style="list-style-type: none"> • O token de confiança é um ICRX assinado com um certificado X.509. <p>O CICS resolve o certificado X.509 para um ID do usuário e assegura que a assinatura XML seja válida. O CICS resolve a identidade baseada em ICRX declarada para um ID do usuário. O CICS, então, executa uma verificação de segurança substituta a partir da identidade X.509 autenticada na identidade ICRX declarada.</p> <ul style="list-style-type: none"> • O token de identidade é um token do nome de usuário sem uma senha. O CICS coloca o nome do usuário no contêiner DFHWS-USERID e o ICRX no contêiner DFHWS-ICRX.
assinatura	basic-kerberos	<i>Combinação inválida de valores de atributo</i>
assinatura	assinatura	As mensagens de entrada devem utilizar a identidade asserida: <ul style="list-style-type: none"> • O token de confiança é um certificado X.509 • O token de identidade é um segundo certificado X.509. O CICS coloca o ID do usuário associado a este certificado no contêiner DFHWS-USERID. <p>O token de identidade e o corpo devem ser assinados com o primeiro certificado X.509 (o token de confiança).</p>

Notas:

1. As combinações dos valores de atributos trust e mode são verificadas quando o ENFILEIRAMENTO é instalado. A instalação falhará se os atributos forem codificados incorretamente.
2. O CICS usa a verificação de senha para verificar um ID do usuário durante os processos descritos no VERIFY PHRASE.

Contém:

1. Um elemento <suppress/> vazio, opcional.
Se este elemento for especificado em um pipeline do provedor de serviços, o manipulador não tentará usar nenhum token de segurança na mensagem para determinar sob qual ID do usuário o serviço é executado.
Se este elemento for especificado em um pipeline do solicitante de serviço, o manipulador não tentará incluir na mensagem SOAP de saída qualquer um dos tokens de segurança que são necessários para autenticação.
2. Em um pipeline do solicitante, um elemento <algorithm> opcional que especifica o URI do algoritmo que é usado para assinar o corpo da mensagem SOAP. Você deve especificar este elemento se a combinação de valores de atributo de confiança e modo indicar que as mensagens foram assinadas. É possível especificar somente o RSA com o algoritmo SHA1 neste elemento. O URI é <http://www.w3.org/2000/09/xmldsig#rsa-sha1>.
3. Um elemento <certificate_label> opcional que especifica a etiqueta que está associada a um certificado digital X.509 instalado no RACF. Se especificar este elemento em um pipeline do solicitante de serviço e o elemento <suppress> não for especificado, o certificado será incluído no cabeçalho de segurança na mensagem SOAP. Se você não especificar um elemento <certificate_label>, o CICS utilizará o certificado padrão no anel de chaves do RACF.
Este elemento é ignorado em um enfileiramento do provedor de serviços.

Exemplo

```
<authentication trust="signature" mode="basic">  
  <suppress/>  
  <algorithm>http://www.w3.org/2000/09/xmldsig#rsa-sha1</algorithm>  
  <certificate_label>AUTHCERT03</certificate_label>  
</authentication>
```

O elemento <sts_authentication>:

Especifica que um Security Token Service (STS) deve ser usado para autenticação e determina qual tipo de solicitação é enviada.

Utilizado em:

- Provedor de Serviços
- Solicitante de serviços

Contido por:

```
<dfhwsse_configuration>
```

Atributos:

Nome	Descrição
Ação	<p>Especifica qual tipo de solicitação o CICS envia ao STS quando uma mensagem é recebida no pipeline do provedor de serviços. Os valores válidos são os seguintes:</p> <p>emissão O STS emite um token de identidade para a mensagem SOAP. Esse valor não é válido para SAML em um pipeline do provedor.</p> <p>validate O STS valida o token de identidade fornecido e retorna se o token for válido para o manipulador de segurança. Se você não especificar esse atributo, o CICS assumirá que a ação é solicitar um token de identidade.</p> <p>Em um pipeline do solicitante de serviço, não é possível especificar este atributo porque o CICS sempre solicita que o STS emita um token.</p>
extract	<p>Este atributo é válido somente quando você está usando SAML. Os elementos do token SAML devem ser extraídos? Os valores válidos são os seguintes:</p> <p>não Os elementos do token SAML não são extraídos para contêineres.</p> <p>sim Os principais elementos do token SAML são extraídos e colocados em contêineres que são criados pelo CICS.</p>
token_signature	<p>Este atributo é válido somente quando você está usando SAML. Uma assinatura de token deve ser fornecida? Os valores válidos são os seguintes:</p> <p>ignored Qualquer assinatura fornecida é ignorada.</p> <p>Requerido Uma assinatura válida deve ser fornecida. Esse é o valor padrão.</p>

Nome	Descrição
tran_channel	<p>Este atributo é válido somente quando você está usando SAML. Em um pipeline do provedor de serviços, este atributo especifica se as asserções SAML contidas em uma mensagem que é recebida no pipeline são disponibilizadas para o programa aplicativo de destino em contêineres no canal de transação DFHTRANSACTION. Os valores válidos são os seguintes:</p> <p>sim As asserções SAML são copiadas nos contêineres no canal DFHTRANSACTION para ser disponibilizadas no programa. Para obter mais informações sobre nomes e tipos de contêiner, consulte .</p> <p>não As asserções SAML não são disponibilizadas para o programa por meio do canal DFHTRANSACTION, mas em contêineres no canal que é transmitido ao programa pelo pipeline. Esse é o valor padrão.</p> <p>Se você não especificar este atributo para um provedor de serviços, as asserções serão disponibilizadas somente em contêineres no canal que é transmitido ao programa a partir do pipeline SOAP.</p> <p>Em um pipeline do solicitante de serviço, este atributo especifica se o token SAML contido no contêiner DFHSAML-OUTTOKEN do canal de transação DFHTRANSACTION é usado na solicitação. Os valores válidos são os seguintes:</p> <p>sim O conteúdo do contêiner DFHSAML-OUTTOKEN do canal DFHTRANSACTION é usado como o token SAML para a solicitação.</p> <p>não O conteúdo do contêiner DFHSAML-OUTTOKEN no canal que é transmitido ao pipeline é usado como o token SAML da solicitação. Esse é o valor padrão.</p> <p>Se você não especificar este atributo para um solicitante de serviço, o token SAML será obtido a partir do contêiner DFHSAML-OUTTOKEN no canal que é transmitido ao pipeline SOAP.</p>

Contém:

- Um elemento <auth_token_type>. Este elemento é necessário quando você especifica um elemento <sts_authentication> em um pipeline do solicitante de serviço e é ideal em um pipeline do provedor de serviços. Para obter mais informações, consulte <auth_token_type>.
 - Em um pipeline do solicitante de serviço, o elemento <auth_token_type> indica o tipo de token que o STS emite quando o CICS envia o ID do usuário contido no contêiner DFHWS-USERID. O token que o CICS recebe do STS é colocado no cabeçalho da mensagem de saída.
 - Em um pipeline do provedor de serviços, o elemento <auth_token_type> é usado para determinar o token de identidade que o CICS obtém do cabeçalho da mensagem e envia ao STS para trocar ou validar. O CICS usa o primeiro token de identidade do tipo especificado no cabeçalho da mensagem. Se você não especificar este elemento, o CICS usará o primeiro

token de identidade que ele localizar no cabeçalho da mensagem. O CICS não considera o seguinte como tokens de identidade:

- wsu:Timestamp
- xenc:ReferenceList
- xenc:EncryptedKey
- ds:Signature

2. Somente em um enfileiramento de provedor de serviços, um elemento `<suppress/>` opcional e vazio. Se esse elemento for especificado, o manipulador não tentará usar nenhum token de segurança na mensagem para determinar o ID do usuário sob o qual o serviço é executado. O elemento `<suppress/>` inclui o token de identidade que é retornado pelo STS.

Exemplo

O seguinte exemplo mostra um enfileiramento do provedor de serviços, no qual o manipulador de segurança requer um token do STS.

```
<sts_authentication action="issue">
  <auth_token_type>
    <namespace>http://example.org.tokens</namespace>
    <element>UsernameToken</element>
  </auth_token_type>
  <suppress/>
</sts_authentication>
```

O elemento `<auth_token_type>`:

Especifica qual tipo de token de identidade é necessário.

Esse elemento é obrigatório ao especificar o elemento `<sts_authentication>` em um enfileiramento do solicitante de serviços e opcional em um provedor de serviços.

- Em um pipeline do solicitante de serviço, o elemento `<auth_token_type>` indica o tipo de token que o STS emite quando o CICS envia o ID do usuário contido no contêiner DFHWS-USERID. O token que o CICS recebe do STS é colocado no cabeçalho da mensagem de saída.
- Em um pipeline do provedor de serviços, o elemento `<auth_token_type>` é usado para determinar o token de identidade que o CICS obtém do cabeçalho da mensagem e envia ao STS para trocar ou validar. O CICS usa o primeiro token de identidade do tipo especificado no cabeçalho da mensagem. Se você não especificar este elemento, o CICS usará o primeiro token de identidade que ele localizar no cabeçalho da mensagem. O CICS não considera o seguinte como tokens de identidade:
 - wsu:Timestamp
 - xenc:ReferenceList
 - xenc:EncryptedKey
 - ds:Signature

Utilizado em:

- Provedor de Serviços
- Solicitante de serviços

Contido por:

`<sts_authentication>`

Contém:

1. Um elemento <namespace>. Esse elemento contém o namespace do tipo de token que deve ser validado ou trocado.
Se você estiver usando SAML, configure o conteúdo desse elemento como urn:oasis:names:tc:SAML:1.0:assertion ou urn:oasis:names:tc:SAML:2.0:assertion, dependendo da versão do SAML.
2. Um elemento <element>. Este elemento contém o nome local do tipo de token que deve ser validado ou trocado.
Para SAML, use a Asserção do nome local.

Os valores desses elementos formam o QName do token.

Exemplo

```
<auth_token_type>  
  <namespace>http://example.org.tokens</namespace>  
  <element>UsernameToken</element>  
</auth_token_type>
```

O elemento <sts_endpoint>:

Especifica o local do Security Token Service (STS).

Utilizado em:

- Provedor de Serviços
- Solicitante de serviços

Contido por:

```
<dfhsse_configuration>
```

Contém:

- Um elemento <endpoint>. Esse elemento contém um URI que aponta para o local do STS (Security Token Service) na rede. É recomendável utilizar o SSL ou o TLS para manter a conexão com o STS segura, em vez de utilizar o HTTP.
Para usar o suporte SAML, configure o terminal como cics://PROGRAM/DFHSAML. Também é possível especificar um terminal do WebSphere MQ usando o formato JMS do URI.
- Um elemento <jvmserver> opcional. Esse elemento identifica o servidor JVM que está configurado para executar o serviço de token SAML. Se esse elemento não for incluído, o servidor JVM do recurso de amostra padrão DFHXSTS será assumido. Esse elemento é válido somente se você estiver usando SAML: se você usá-lo em outras situações, ocorrerá um erro.

Exemplos

Neste exemplo, o terminal é configurado para usar uma conexão segura com o STS no URI especificado.

```
<sts_endpoint>  
  <endpoint>https://example.com/SecurityTokenService</endpoint>  
</sts_endpoint>
```

Neste exemplo, o terminal é configurado para usar o suporte CICS SAML.

```
<sts_endpoint>  
  <endpoint>cics://PROGRAM/DFHSAML</endpoint>  
</sts_endpoint>
```

O Elemento <sign_body>:

Direciona DFHWSSE para assinar o corpo de mensagens SOAP de saída e fornece informações sobre como as mensagens devem ser assinadas.

Utilizado em:

- Provedor de Serviços
- Solicitante de serviços

Contido por:

<dfhwsse_configuration>

Contém:

1. Um elemento <algorithm> que contém o URI que identifica o algoritmo usado para assinar o corpo da mensagem SOAP. É possível especificar os algoritmos mostrados em “Algoritmos de Assinatura” na página 610.
2. Um elemento <certificate_label> que especifica a etiqueta associada a um certificado digital instalado no RACF. O certificado digital fornece a chave que é usada para assinar a mensagem.

Exemplo

```
<sign_body>
  <algorithm>http://www.w3.org/2000/09/xmldsig#rsa-sha1</algorithm>
  <certificate_label>SIGCERT01</certificate_label>
</sign_body>
```

O Elemento <encrypt_body>:

Direciona o DFHWSSE a criptografar o corpo de mensagens SOAP de saída e fornece informações sobre como as mensagens devem ser criptografadas.

Utilizado em:

- Provedor de Serviços
- Solicitante de serviços

Contido por:

<dfhwsse_configuration>

Contém:

1. Um elemento <algorithm> contendo o URI que identifica o algoritmo usado para criptografar o corpo da mensagem SOAP. É possível especificar os algoritmos mostrados em “Algoritmos de Criptografia” na página 612.
2. Um elemento <certificate_label> que especifica a etiqueta que está associada a um certificado digital no RACF. O certificado digital fornece a chave que é usada para criptografar a mensagem.

Exemplo

```
<encrypt_body>
  <algorithm>http://www.w3.org/2001/04/xmlenc#aes256-cbc</algorithm>
  <certificate_label>ENCCERT02</certificate_label>
</encrypt_body>
```

Manipuladores de Aplicativo

Um manipulador de aplicativo é um programa CICS que o manipulador de terminal de um pipeline do provedor de serviços SOAP vincula ao tempo de execução.

Os manipuladores de aplicativo são usados em pipelines do modo de provedor nos quais o manipulador de terminal é um dos manipuladores de mensagem SOAP fornecidos. Esta situação ocorre quando o elemento `<terminal_handler>` contém um elemento `<cics_soap_1.1_handler>`, `<cics_soap_1.2_handler>`, `<cics_soap_1.1_handler_java>` ou `<cics_soap_1.2_handler_java>`.

O manipulador de aplicativo é responsável por processar o corpo de uma solicitação SOAP e por gerar uma resposta usando os dados retornados. O manipulador de aplicativo pode chamar outros programas para concluir este processamento. Geralmente o manipulador de aplicativo age como uma camada de apresentação de propósito geral em torno de um ou mais aplicativos de negócios. Ele é responsável por mapear XML para um formulário que um aplicativo pode usar, conectando esse aplicativo e, em seguida, gerando uma resposta usando os dados retornados.

Um manipulador de aplicativo pode ser conectado pelo CICS de duas maneiras. O mecanismo típico envolve contêineres de canal e de controle; o outro método envolve ligações Java para Axis2.

Os manipuladores de aplicativo conectados ao canal são especificados no elemento `<apphandler>` do elemento `<provider_pipeline>`. No tempo de execução, o contêiner DFHWS-APPHANDLER é preenchido pelo conteúdo de `<apphandler>`. No entanto, o contêiner DFHWS-APPHANDLER pode ser atualizado dinamicamente por qualquer um dos outros manipuladores de mensagem. Portanto, o programa que é vinculado ao tempo de execução pode ser diferente para o programa especificado no elemento `<apphandler>`. Os manipuladores de aplicativo a seguir podem ser especificados no elemento `<apphandler>` ou no contêiner DFHWS-APPHANDLER:

- O manipulador de aplicativo SOAP conectado ao canal fornecido, DFHPITP. Para obter mais informações sobre manipuladores de aplicativo conectados ao canal, consulte “Manipuladores de Aplicativo Conectados ao Canal” na página 133
- Seu próprio manipulador de aplicativo conectado ao canal. Este manipulador de aplicativo pode ser gravado em idiomas diferentes de Java. Para obter mais informações sobre os contêineres de controle que podem ser usados em seu manipulador de aplicativo conectado ao canal, consulte “Contêineres de Controle” na página 150.
- Seu próprio manipulador de aplicativo Java para pipelines baseados em Java, que implementa a interface Java ApplicationHandler e que está conectado ao pipeline usando MessageContext do Axis2. Para obter mais informações sobre a interface Java ApplicationHandler, consulte Referência de Classe JCICS.

Para usar um manipulador de aplicativo que usa ligações Java para Axis2, você deve especificar o elemento `<apphandler_class>` do elemento `<provider_pipeline>`. Os manipuladores de aplicativo Axis2 também requerem que um servidor de JVM exista para o pipeline de serviços da web e o manipulador de aplicativo para ser executado e que o manipulador terminal de seu pipeline de serviços da web deve ser o manipulador de mensagem `<cics_soap_1.1_handler_java>` ou `<cics_soap_1.2_handler_java>`. Para usar o manipulador de aplicativo Axis2 fornecido, você deve especificar `com.ibm.cicsts.axis2.CICSAxis2ApplicationHandler` no elemento

<apphandler_class>, no entanto, é possível especificar sua própria classe do manipulador de aplicativo Axis2. No tempo de execução, o contêiner DFHWS-APPHANCLAS é preenchido pelo conteúdo de <apphandler_class>.

Para aplicativos de serviço da web que são implementados usando o assistente de serviços da web do CICS, você deve especificar DFHPITP ou seu próprio manipulador de aplicativo que usa DFHPITP no elemento <apphandler> ou especificar com.ibm.cicsts.axis2.CICSAXIS2ApplicationHandler no elemento <apphandler_class>. Para obter mais informações sobre o assistente de serviços da web do CICS, consulte O Assistente de Serviços da Web CICS.

Também é possível implementar aplicativos Axis2 como serviços da web no modo de provedor no CICS usando o estilo Axis2 de implementação de serviço da web. Para obter mais informações, consulte Implementando um serviço da web no modo de provedor Java em um servidor JVM Axis2.

Manipuladores de Aplicativo Conectados ao Canal

Os manipuladores de aplicativo conectados ao canal são manipuladores de aplicativo que são anexados ao CICS usando um canal e contêineres de controle.

O canal que é usado pelo manipulador de aplicativo é o canal DFHAHC-V1. Este canal transmite os seguintes contêineres entre o manipulador de terminal e o aplicativo de serviço da web do modo de provedor:

DFHWS-XMLNS

Contém uma lista de pares nome-valor que mapeiam prefixos de namespace para namespaces.

- Na entrada, a lista contém os namespaces que estão no escopo do envelope SOAP.
- Na saída, a lista contém os dados de namespace que são assumidos como estando na tag de envelope.

DFHWS-BODY

Contém a seção do corpo do envelope SOAP. Geralmente, o aplicativo modificará o conteúdo. Se o aplicativo não modificar o conteúdo, o programa do manipulador de aplicativo deverá atualizar o conteúdo deste contêiner, mesmo se ele estiver colocando o mesmo conteúdo de volta no contêiner antes de retornar ao manipulador terminal.

DFHNORESPONSE

Na fase de solicitação de um pipeline do solicitante de serviço, indica que não espera-se que o provedor de serviços retorne uma resposta. O conteúdo do contêiner DFHNORESPONSE é indefinido; os manipuladores de mensagem que precisam saber se o provedor de serviços deve retornar uma resposta precisam somente determinar se o contêiner está presente ou não:

- Se o contêiner DFHNORESPONSE estiver presente, nenhuma resposta é esperada.
- Se o contêiner DFHNORESPONSE estiver ausente, uma resposta é esperada.

O canal também transmite todos os contêineres de contextos que foram transmitidos ao manipulador de terminal. Por exemplo, um programa de processamento de cabeçalho pode incluir contêineres no canal. Estes contêineres são transmitidos como contêineres do usuário. Para obter mais informações sobre manipuladores de aplicativo, consulte “Manipuladores de Aplicativo” na página 90.

Manipuladores de mensagens

Um manipulador de mensagem é um programa CICS que é utilizado para processar uma solicitação de serviço da Web durante a entrada e para processar a resposta durante a saída. Os manipuladores de mensagens usam canais e contêineres para interagir uns com os outros e com o sistema.

A interface do manipulador de mensagem permite executar as tarefas a seguir em um programa do manipulador de mensagem:

- Examine o conteúdo de uma solicitação ou resposta XML ou JSON, sem mudá-lo
- Mude o conteúdo de uma solicitação ou resposta XML ou JSON
- Em um manipulador de mensagem não de terminal, transmita uma solicitação ou resposta XML ou JSON para o próximo manipulador de mensagem no pipeline
- Em um manipulador de mensagem do terminal, chame um programa de aplicativo e gere uma resposta
- Na fase de solicitação do pipeline, force uma transição para a fase de resposta, absorvendo a solicitação e gerando uma resposta
- Manipular Erros

Dica: É aconselhável usar manipuladores SOAP, `<cics_soap_1.1_handler>`, `<cics_soap_1.2_handler>`, `<cics_soap_1.1_handler_java>` ou `<cics_soap_1.2_handler_java>`, para trabalhar com mensagens SOAP. Estes manipuladores permitem trabalhar diretamente com os principais elementos em uma mensagem SOAP (os manipuladores SOAP e o corpo SOAP).

Todos os programas que são usados como manipuladores de mensagem são chamados com a mesma interface: eles são chamados com um *canal* que contém vários contêineres. Os contêineres podem ser categorizados como os tipos a seguir:

Contêineres de Controle

Eles são essenciais para a operação do pipeline. Os manipuladores de mensagem podem usar os contêineres de controle para modificar a sequência na qual manipuladores subsequentes são processados.

Contêineres de Contextos

Em algumas situações, os programas do manipulador de mensagem precisam de informações sobre o contexto no qual eles são chamados. O CICS fornece estas informações em um conjunto de *contêineres de contextos* que são transmitidos aos programas.

Alguns dos contêineres de contextos contêm informações que podem ser alteradas em seu manipulador de mensagem. Por exemplo, em um pipeline do provedor de serviços, é possível alterar o ID do usuário e o ID da transação do programa do aplicativo de destino modificando o conteúdo dos contêineres de contextos apropriados.

Contêineres do Usuário

Eles contêm informações que um manipulador de mensagem precisa transmitir para um outro. O uso de contêineres de usuário é inteiramente uma questão para os manipuladores de mensagens.

Restrição: Não use nomes que iniciam com DFH para contêineres do usuário.

Como Contêineres Controlam os Protocolos de Pipeline

Os conteúdos dos contêineres DFHFUNCTION, DFHREQUEST e DFHRESPONSE juntos controlam a protocolos de pipeline.

Durante as duas fases da execução de um pipeline (a fase de solicitação e a fase de resposta) o valor de DFHFUNCTION determina quais contêineres de controle são transmitidos para cada manipulador de mensagem:

DFHFUNCTION	Contexto	DFHREQUEST	DFHRESPONSE
RECEIVE-REQUEST	Provedor de serviços; fase de solicitação	Presente (comprimento > 0)	Presente (comprimento = 0)
SEND-RESPONSE	Provedor de serviços; fase de resposta	Ausente	Presente (comprimento > 0)
SEND-REQUEST	Solicitante de serviço; fase de solicitação	Presente (comprimento > 0)	Presente (comprimento = 0)
RECEIVE-RESPONSE	Solicitante de serviço; fase de resposta	Ausente	Presente (comprimento > 0)
PROCESS-REQUEST	Provedor de serviços; manipulador de terminal	Presente (comprimento > 0)	Presente (comprimento = 0)
HANDLER-ERROR	Solicitante ou provedor de serviço; qualquer fase	Ausente	Presente (comprimento = 0)
NO-RESPONSE	Solicitante ou provedor de serviço; fase de resposta	Ausente	Ausente

o processamento subsequente é determinado pelo contêineres que seu manipulador de mensagens transmite de volta para o pipeline:

Durante a fase de solicitação

- Seu manipulador de mensagem pode retornar o contêiner DFHREQUEST. O processamento continua na fase de solicitação com o próximo manipulador. O comprimento dos dados no contêiner não deve ser zero.
- Seu manipulador de mensagem pode retornar o contêiner DFHRESPONSE. O processamento alterna para a fase de resposta e o mesmo manipulador é chamado com DFHFUNCTION configurado como SEND-RESPONSE em um provedor de serviços e com RECEIVE-RESPONSE em um solicitante de serviço. O comprimento dos dados no contêiner não deve ser zero.
- Seu manipulador de mensagem pode não retornar nenhum contêiner. O processamento alterna para a fase de resposta e o mesmo manipulador é chamado com DFHFUNCTION configurado como NO-RESPONSE.

No manipulador terminal (somente provedor de serviços)

- Seu manipulador de mensagem pode retornar o contêiner DFHRESPONSE. O processamento alterna para a fase de resposta e o manipulador anterior é chamado com um novo valor igual a DFHFUNCTON (SEND-RESPONSE). O comprimento dos dados no contêiner não deve ser zero.
- Seu manipulador de mensagem pode não retornar nenhum contêiner. O processamento alterna para a fase de resposta e o manipulador anterior é chamado com um novo valor igual a DFHFUNCTON (NO-RESPONSE).

Durante a fase de resposta

- Seu manipulador de mensagem pode retornar o contêiner DFHRESPONSE. O processamento continua na fase de resposta e o próximo manipulador é chamado. O comprimento dos dados no contêiner não deve ser zero.
- Seu manipulador de mensagem pode não retornar nenhum contêiner. O processamento continua na fase de resposta e o próximo manipulador na sequência é chamado com um novo valor igual a DFHFUNCTON (NO-RESPONSE).

Importante: Durante a fase de solicitação, seu manipulador de mensagem pode retornar DFHREQUEST ou DFHRESPONSE, mas não ambos. Como ambos os contêineres estão presentes quando seu manipulador de mensagem é chamado, você deve excluir um deles.

Esta tabela mostra a ação executada pelo pipeline para todos os valores de DFHFUNCTON e todas as combinações de DFHREQUEST e DFHRESPONSE retornadas por cada manipulador de mensagens.

DFHFUNCTON	Contexto	DFHREQUEST	DFHRESPONSE	Ação
RECEIVE-REQUEST	Provedor de serviços; fase de solicitação	Presente (comprimento > 0)	Presente	(erro)
RECEIVE-REQUEST	Provedor de serviços; fase de solicitação	Presente (comprimento > 0)	Ausente	Chame o próximo manipulador com a função RECEIVE-REQUEST
RECEIVE-REQUEST	Provedor de serviços; fase de solicitação	Presente (comprimento = 0)	Não aplicável	(erro)
RECEIVE-REQUEST	Provedor de serviços; fase de solicitação	Ausente	Presente (comprimento > 0)	Altere para a fase de resposta e chame o mesmo manipulador com a função SEND-RESPONSE
RECEIVE-REQUEST	Provedor de serviços; fase de solicitação	Ausente	Presente (comprimento = 0)	(erro)
RECEIVE-REQUEST	Provedor de serviços; fase de solicitação	Ausente	Ausente	Chame o mesmo manipulador com a função NO-RESPONSE
SEND-RESPONSE	Provedor de serviços; fase de resposta	Não aplicável	Presente (comprimento > 0)	Chame o manipulador anterior com a função SEND-RESPONSE
SEND-RESPONSE	Provedor de serviços; fase de resposta	Não aplicável	Presente (comprimento = 0)	(erro)
SEND-RESPONSE	Provedor de serviços; fase de resposta	Não aplicável	Ausente	Chame o mesmo manipulador com a função NO-RESPONSE

DFHFUNCTION	Contexto	DFHREQUEST	DFHRESPONSE	Ação
SEND-REQUEST	Solicitante de serviço; fase de solicitação	Presente (comprimento > 0)	Presente (comprimento ≥ 0)	(erro)
SEND-REQUEST	Solicitante de serviço; fase de solicitação	Presente (comprimento > 0)	Ausente	Chame o próximo manipulador com a função SEND-REQUEST
SEND-REQUEST	Solicitante de serviço; fase de solicitação	Presente (comprimento = 0)	Não aplicável	(erro)
SEND-REQUEST	Solicitante de serviço; fase de solicitação	Ausente	Presente (comprimento > 0)	Altere para a fase de resposta e chame o manipulador anterior com a função RECEIVE-RESPONSE
SEND-REQUEST	Solicitante de serviço; fase de solicitação	Ausente	Presente (comprimento = 0)	(erro)
SEND-REQUEST	Solicitante de serviço; fase de solicitação	Ausente	Ausente	Chame o mesmo manipulador com a função NO-RESPONSE
RECEIVE-RESPONSE	Solicitante de serviço; fase de resposta	Não aplicável	Presente (comprimento > 0)	Chame o manipulador anterior com a função RECEIVE-RESPONSE
RECEIVE-RESPONSE	Solicitante de serviço; fase de resposta	Não aplicável	Presente (comprimento = 0)	(erro)
RECEIVE-RESPONSE	Solicitante de serviço; fase de resposta	Não aplicável	Ausente	Chame o mesmo manipulador com a função NO-RESPONSE
PROCESS-REQUEST	Provedor de serviços; manipulador de terminal	Não aplicável	Presente (comprimento > 0)	Chame o manipulador anterior com a função RECEIVE-RESPONSE
PROCESS-REQUEST	Provedor de serviços; manipulador de terminal	Não aplicável	Presente (comprimento = 0)	(erro)
PROCESS-REQUEST	Provedor de serviços; manipulador de terminal	Não aplicável	Ausente	Chame o mesmo manipulador com a função NO-RESPONSE
HANDLER-ERROR	Solicitante ou provedor de serviço; qualquer fase	Não aplicável	Presente (comprimento > 0)	Chame o manipulador anterior com a função SEND-RESPONSE ou a função RECEIVE-RESPONSE
HANDLER-ERROR	Solicitante ou provedor de serviço; qualquer fase	Não aplicável	Presente (comprimento = 0)	(erro)
HANDLER-ERROR	Solicitante ou provedor de serviço; qualquer fase	Não aplicável	Ausente	Chame o mesmo manipulador com a função NO-RESPONSE

Fornecendo seus Próprios Manipuladores de Mensagens

Quando desejar executar o processamento especializado nas mensagens que fluem entre um solicitante de serviço e um provedor de serviços, e o CICS não fornecer um manipulador de mensagem que atende às suas necessidades, você precisará fornecer seu próprio.

Sobre Esta Tarefa

Na maioria das situações, é possível executar todo o processamento necessário com os manipuladores de mensagens fornecidos com o CICS. Por exemplo, é possível usar os manipuladores de mensagem SOAP 1.1 e 1.2 que o CICS fornece para processar mensagens SOAP. Mas há ocasiões quando você desejará executar suas próprias operações especializadas nas solicitações e respostas de serviço da web. Para fazer isto, você deve fornecer seus próprios manipuladores de mensagens.

Procedimento

1. Grave seu programa manipulador de mensagem. Um manipulador de mensagem é um programa CICS com uma interface do canal. É possível gravar seu programa em qualquer um dos idiomas que o CICS suporta e usar qualquer comando CICS no subconjunto DPL dentro de seu programa.
2. Compile e edite o link de seu programa. Os programas manipuladores de mensagens normalmente são executados sob a transação CPIH, que é definida com o atributo TASKDATALOC(ANY). Portanto, quando você edita o link do programa, deve especificar a opção AMODE(31).
3. Instale o programa em seu sistema CICS da maneira usual.
4. Defina o programa no arquivo de configuração de pipeline. Use o elemento `<handler>` para definir seu manipulador de mensagem. Dentro do elemento `<handler>`, codifique um elemento `<program>` contendo o nome do programa.

Trabalhando com Mensagens em um Manipulador de Mensagem Não de Terminal

Um manipulador de mensagem não de terminal típico processa uma mensagem, em seguida, passa o controle para um outro manipulador de mensagem no pipeline.

Sobre Esta Tarefa

Em um manipulador de mensagem não de terminal, é possível trabalhar com uma solicitação ou resposta, alterando-a ou não, e transmiti-la para o próximo manipulador de mensagem.

Nota: Embora os serviços da web geralmente usem mensagens SOAP que contêm XML, seus manipuladores de mensagem funcionarão também com outros formatos da mensagem

Procedimento

1. Utilizando o conteúdo do contêiner DFHFUNCTION, determine se a mensagem transmitida para esse manipulador de mensagem é uma solicitação ou uma resposta.

DFHFUNCTION	Solicitação ou Resposta	Tipo de Manipulador de Mensagem	Entrada ou Saída
RECEIVE-REQUEST	Request	Não de terminal	Entrada
SEND-RESPONSE	respostas	Não de terminal	Transmissão
SEND-REQUEST	Request	Não de terminal	Transmissão
RECEIVE-RESPONSE	respostas	Não de terminal	Entrada

Dica:

- Se DFHFUNCTION contiver PROCESS-REQUEST, o manipulador de mensagem é um manipulador de mensagem de terminal e estas etapas não se aplicam.
 - Se DFHFUNCTION contiver HANDLER-ERROR, o manipulador está sendo chamado para processamento de erro e estas etapas não se aplicam.
2. Recupere a solicitação ou a resposta do contêiner apropriado.
 - Se a mensagem for uma solicitação, ela será transmitida ao programa no contêiner DFHREQUEST. O contêiner DFHRESPONSE também está presente, com um comprimento igual a zero.
 - Se a mensagem for uma resposta, ela será transmitida ao programa no contêiner DFHRESPONSE.
 3. Execute qualquer processamento da mensagem que seja necessário. Dependendo do propósito do manipulador de mensagem, você pode:
 - Examinar a mensagem sem alterá-la e transmiti-la para o próximo manipulador de mensagem no pipeline.
 - Alterar a solicitação e transmiti-la para o próximo manipulador de mensagem no pipeline.
 - Se a mensagem for uma solicitação, será possível efetuar bypass nos manipuladores de mensagem a seguir no pipeline e, em substituição, construir uma mensagem de resposta.

Nota: É o conteúdo dos contêineres que um manipulador de mensagem retorna que determina qual manipulador de mensagem é chamado em seguida.

É um erro se um manipulador de mensagem não faz mudanças em nenhum dos contêineres transmitidos a ele.

É um erro para um programa do manipulador de mensagem retornar qualquer um dos seguintes:

- Um contêiner DFHRESPONSE vazio.
- Um contêiner DFHREQUEST não vazio e um contêiner DFHRESPONSE não vazio.
- Um contêiner DFHREQUEST vazio na solicitação de saída.

Transmitindo uma Mensagem para o Próximo Manipulador de Mensagem no Pipeline:

Em um manipulador de mensagem não de terminal típico, você processará uma solicitação ou resposta, alterando-a ou não, e a transmitirá para o próximo manipulador de mensagem.

Procedimento

1. Retorne a mensagem ao pipeline - alterada ou inalterada - no contêiner apropriado.
 - Se a mensagem for uma solicitação e você a tiver alterado, retorne-a no contêiner DFHREQUEST
 - Se a mensagem for uma resposta e você a tiver alterado, coloque-a no contêiner DFHRESPONSE
 - Se você não tiver alterado a mensagem, ela já está no contêiner apropriado
2. Se a mensagem for uma solicitação, exclua o contêiner DFHRESPONSE. Quando um manipulador de mensagem é chamado para uma solicitação, os contêineres DFHREQUEST e DFHRESPONSE são transmitidos ao programa;

DFHRESPONSE possui um comprimento igual a zero. No entanto, é um erro retornar DFHREQUEST e DFHRESPONSE.

Resultados

A mensagem é transmitida ao próximo manipulador de mensagem no pipeline.

Forçando uma Transição para a Fase de Resposta do Pipeline:

Quando estiver processando uma solicitação, haverá vezes em que você desejará gerar uma resposta imediata, em vez de transmitir a solicitação ao próximo manipulador de mensagem no pipeline.

Procedimento

1. Exclua o contêiner DFHREQUEST.
2. Construa sua resposta e coloque-a no contêiner DFHRESPONSE.

Resultados

A resposta é transmitida ao próximo manipulador de mensagem na fase de resposta do pipeline.

Suprimindo a Resposta:

Em algumas situações, você desejará absorver uma solicitação sem enviar uma resposta.

Procedimento

1. Exclua o contêiner DFHREQUEST.
2. Exclua o contêiner DFHRESPONSE.

Manipulando Mensagens Unidirecionais em um Pipeline do Solicitante de Serviço:

Quando um pipeline do solicitante de serviço precisa de uma solicitação para um provedor de serviços, normalmente há uma expectativa de que haja uma resposta e que, após o envio da solicitação, os manipuladores de mensagem no pipeline sejam chamados novamente quando a resposta chegar. Alguns serviços da web não enviam uma resposta e, portanto, você deve executar uma ação especial para indicar que o CICS não deve aguardar por uma resposta antes de chamar os manipuladores de mensagens pela segunda vez.

Sobre Esta Tarefa

Para fazer isso, verifique se o contêiner DFHNORESPONSE está presente no final do processamento de pipeline na fase de solicitação. Geralmente, isso é feito pelo código no nível do aplicativo, porque o conhecimento de se uma resposta é esperada é registrado no aplicativo:

- Para aplicativos implementados com o assistente de serviços da web do CICS, o código do CICS criará o contêiner.
- Os aplicativos que não são implementados com o assistente geralmente criarão o contêiner antes de chamar o aplicativo.

Se criar ou destruir o contêiner DFHNORESPONSE em um manipulador de mensagem, você deverá ter certeza de que isso não irá comprometer o protocolo de

mensagens entre o solicitante e o provedor de serviços.

Trabalhando com Mensagens em um Manipulador de Mensagem do Terminal

Um manipulador de terminal típico processa uma solicitação, chama um programa de aplicativo e gera uma resposta.

Sobre Esta Tarefa

Nota: Embora os serviços da web geralmente usem mensagens SOAP que contêm XML, seus manipuladores de mensagem funcionarão também com outros formatos da mensagem

Em um manipulador de mensagem do terminal, você pode trabalhar com uma solicitação e, opcionalmente, gerar uma resposta e transmiti-la de volta ao longo do pipeline. Um manipulador de terminal típico utilizará a solicitação como entrada para um programa de aplicativo e utilizará a resposta do programa de aplicativo para construir a resposta.

Procedimento

1. Usando o conteúdo do contêiner DFHFUNCTION, determine se a mensagem transmitida para este manipulador é uma solicitação e se o manipulador está sendo chamado como um manipulador de terminal.

DFHFUNCTION	Solicitação ou Resposta	Tipo de Manipulador	Entrada ou Saída
PROCESS-REQUEST	Request	Terminal	Entrada

Dica:

- Se DFHFUNCTION contiver qualquer outro valor, o manipulador não é um manipulador de terminal e estas etapas não se aplicam.
2. Recupere a solicitação do contêiner DFHREQUEST. O contêiner DFHRESPONSE também está presente, com um comprimento igual a zero.
 3. Execute qualquer processamento da mensagem que seja necessário. Geralmente, um manipulador de terminal chamará um programa de aplicativo.
 4. Construa sua resposta e coloque-a no contêiner DFHRESPONSE. Se não houver nenhuma resposta, você deverá excluir o contêiner DFHRESPONSE.

Resultados

A resposta é transmitida ao próximo manipulador na fase de resposta do pipeline. O manipulador é chamado para a função SEND-RESPONSE. Se não houver nenhuma resposta, o próximo manipulador será chamado para a função NO-RESPONSE.

Manipulando erros

Os manipuladores de mensagens devem ser projetados para manipular erros que podem ocorrer no pipeline.

Sobre Esta Tarefa

Quando um erro ocorre em um programa manipulador de mensagem, o programa é chamado novamente para processamento de erro. O processamento de erro

sempre ocorre na fase de resposta do pipeline; se o erro ocorreu na fase de solicitação, manipuladores subsequentes na fase de solicitação são ignorados.

Na maioria dos casos, portanto, você deve gravar seu programa manipulador para manipular quaisquer erros que possam ocorrer.

Procedimento

1. Verifique se o contêiner DFHFUNCTON contém HANDLER-ERROR, indicando que o manipulador de mensagem foi chamado para processamento de erro.

Dica:

- Se DFHFUNCTON contiver qualquer outro valor, o manipulador de mensagens não tiver sido chamado para processamento de erro e estas etapas não se aplicam.
2. Analise as informações de erro e determine se o manipulador de mensagens pode se recuperar do erro construindo uma resposta adequada.
O contêiner DFHERROR contém informações sobre o erro. Para obter informações detalhadas sobre este contêiner, consulte “Contêiner DFHERROR” na página 150.
O contêiner DFHRESPONSE também está presente, com um comprimento igual a zero.
 3. Execute qualquer processamento de recuperação.
 - Se o manipulador de mensagens puder se recuperar, construa uma resposta e retorne-a no contêiner DFHRESPONSE.
 - Se o manipulador de mensagem puder se recuperar, mas nenhuma resposta for necessária, exclua o contêiner DFHRESPONSE e retorne o contêiner DFHNORESPONSE no lugar.
 - Se o manipulador de mensagens não puder se recuperar, retorne o contêiner DFHRESPONSE inalterado (ou seja, com um comprimento zero).

Resultados

Se o seu manipulador de mensagem for capaz de recuperar-se do erro, o processamento de pipeline continuará normalmente. Caso contrário, o CICS gerará uma falha SOAP que contém informações sobre o erro. No caso de um encerramento anormal de transação, o código de encerramento anormal será incluído na falha.

A Interface do Manipulador de Mensagem

O pipeline do CICS vincula aos manipuladores de mensagem usando um canal contendo inúmeros contêineres. Alguns contêineres são opcionais, outros são requeridos por todos os manipuladores de mensagem e outros são usados por alguns manipuladores de mensagem e não por outros.

Antes de um manipulador ser chamado, alguns ou todos os contêineres são preenchidos com informações que o manipulador pode usar para executar seu trabalho. Os contêineres retornados pelo manipulador determinam o processamento subsequente e são transmitidos para manipuladores posteriores no pipeline.

Os Manipuladores de Mensagem SOAP

Os manipuladores de mensagem SOAP são manipuladores de mensagem fornecidos pelo CICS que podem ser incluídos em seu pipeline para processar mensagens SOAP 1.1 e SOAP 1.2. É possível usar os manipuladores de mensagem SOAP em um pipeline do solicitante de serviço ou do provedor de serviços.

Na entrada, os manipuladores de mensagem SOAP analisam mensagens SOAP de entrada e extraem o elemento SOAP <Body> para uso por seu programa de aplicativo. Na saída, os manipuladores constroem a mensagem SOAP completa, usando o elemento <Body> que seu aplicativo fornece.

Se você usar cabeçalhos SOAP em suas mensagens, os manipuladores SOAP poderão chamar programas de processamento de cabeçalho gravados pelo usuário que permitem processar os cabeçalhos em mensagens de entrada e incluí-los em mensagens de saída.

Os manipuladores de mensagem SOAP, e quaisquer programas de processamento de cabeçalho, são especificados no arquivo de configuração de pipeline. Para pipelines que não suportam Java, os manipuladores de mensagem <cics_soap_1.1_handler> ou <cics_soap_1.2_handler> devem ser especificados. Para pipelines que suportam Java, os manipuladores de mensagem <cics_soap_1.1_handler_java> ou <cics_soap_1.2_handler_java> devem ser especificados.

Geralmente, você precisará somente de um manipulador SOAP em um pipeline. No entanto, há algumas situações nas quais mais de um são necessários. Por exemplo, é possível assegurar que os cabeçalhos SOAP sejam processados em uma sequência específica definindo diversos manipuladores SOAP.

Você não deve definir manipuladores de mensagem <cics_soap_1.1_handler> e <cics_soap_1.2_handler> ou manipuladores de mensagem <cics_soap_1.1_handler_java> e <cics_soap_1.2_handler_java> no mesmo pipeline. Se você espera que seu pipeline processe mensagens SOAP 1.1 e SOAP 1.2, deverá usar o manipulador de mensagem <cics_soap_1.2_handler> ou <cics_soap_1.2_handler_java>.

Programas de Processamento de Cabeçalho

Os programas de processamento de cabeçalho são programas CICS gravados pelo usuário que são vinculados a partir dos manipuladores de mensagem SOAP 1.1 e SOAP 1.2 fornecidos pelo CICS, para processar blocos de cabeçalho SOAP.

É possível gravar seu programa de processamento de cabeçalho em qualquer uma das linguagens que o CICS suporta e usar qualquer comando do CICS no subconjunto DPL. Seu programa de processamento de cabeçalho pode vincular-se a outros programas CICS.

Os programas de processamento de cabeçalho possuem uma interface do canal; os contêineres contêm informações que o programa de cabeçalho pode examinar ou modificar, incluindo o bloco de cabeçalho SOAP para o qual o programa é chamado e o corpo da mensagem SOAP.

O canal e os contêineres que o programa de processamento de cabeçalho podem usar são descritos em “A Interface do Programa de Processamento de Cabeçalho” na página 145.

Outros contêineres contêm informações sobre o ambiente no qual o programa de cabeçalho é chamado, por exemplo:

- O ID de transação sob o qual o programa de cabeçalho foi chamado
- Se o programa foi chamado para um provedor de serviços ou um pipeline do solicitante
- Se a mensagem que está sendo processada for uma solicitação ou resposta

Os programas de processamento de cabeçalho normalmente são executados sob a transação CPIH, que é definida com o atributo TASKDATALOC(ANY). Portanto, quando você edita o link do programa, deve especificar a opção AMODE(31).

Como Programas de Processamento de Cabeçalho são Chamados para uma Solicitação SOAP

Os elementos <cics_soap_1.1_handler>, <cics_soap_1.2_handler>, <cics_soap_1.1_handler_java> e <cics_soap_1.2_handler_java> em uma configuração de pipeline contêm zero, um ou mais elementos <headerprogram>, cada um dos quais contém os filhos a seguir:

```
<program_name>
<namespace>
<localname>
<mandatory>
```

Quando um pipeline está processando uma mensagem SOAP de entrada (uma solicitação no caso de um provedor de serviços, uma resposta no caso de um solicitante de serviço), o programa do cabeçalho especificado no elemento <program_name> é chamado ou não, dependendo dos itens a seguir:

- O conteúdo dos elementos <namespace>, <localname> e <mandatory>
- O valor de determinados atributos do elemento-raiz do próprio cabeçalho SOAP (o atributo **actor** para SOAP 1.1; o atributo **role** para SOAP 1.2)

As regras a seguir determinam se o programa de cabeçalho será chamado em um determinado caso:

O elemento <mandatory> no arquivo de configuração de pipeline

Se o elemento contém true (ou 1), o programa de processamento de cabeçalho é chamado pelo menos uma vez, mesmo se nenhum dos cabeçalhos na mensagem SOAP for selecionado para processamento pelas regras restantes:

- Se nenhum dos blocos de cabeçalho forem selecionados, o programa de processamento de cabeçalho será chamado uma vez.
- Se qualquer um dos blocos de cabeçalho for selecionado pelas regras restantes, o programa de processamento de cabeçalho será chamado uma vez para cada cabeçalho selecionado.

Atributos no bloco de cabeçalho SOAP

Para SOAP 1.1, um bloco de cabeçalho é elegível para processamento somente se o atributo **actor** está ausente ou possui um valor igual a <http://schemas.xmlsoap.org/soap/actor/next>

Para SOAP 1.2, um bloco de cabeçalho é elegível para processamento somente se o atributo **role** está ausente ou possui um dos valores a seguir:

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

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

Um bloco de cabeçalho que é elegível para processamento não é processado, a menos que seja selecionado pela próxima regra.

Os elementos <namespace> e <localname> no arquivo de configuração de pipeline

Um bloco de cabeçalho que é elegível para processamento de acordo com a regra anterior é selecionado para processamento somente se as condições a seguir são satisfeitas:

- O nome do elemento-raiz do bloco de cabeçalho corresponde ao elemento <localname> no arquivo de configuração de pipeline
- O namespace do elemento-raiz corresponde ao elemento <namespace> no arquivo de configuração de pipeline

Por exemplo, considere este bloco de cabeçalho:

```
<t:myheaderblock xmlns:t="http://mynamespace" ...> .... </t:myheaderblock>
```

Sujeito a outras regras, o bloco de cabeçalho é selecionado para processamento quando as linhas a seguir são codificadas no arquivo de configuração de pipeline:

```
<namespace>http://mynamespace</namespace>  
<localname>myheaderblock</localname>
```

Os elementos <localname> podem conter um * para indicar que todos os blocos de cabeçalho no namespace devem ser processados. Portanto, o mesmo bloco de cabeçalho é selecionado pelo código a seguir:

```
<namespace>http://mynamespace</namespace>  
<localname>*</localname>
```

Quando a mensagem SOAP contém mais de um cabeçalho, o programa de processamento de cabeçalho é chamado uma vez para cada cabeçalho correspondente, mas a sequência na qual os cabeçalhos são processados é indefinida.

Os manipuladores de mensagem SOAP fornecidos pelo CICS selecionam os programas de processamento de cabeçalho que são chamados com base nos blocos de cabeçalho que estão presentes na mensagem SOAP no momento em que o manipulador de mensagem a recebe. Portanto, um programa de processamento de cabeçalho nunca é chamado como resultado de um bloco de cabeçalho que é incluído em uma mensagem no mesmo manipulador de mensagem SOAP. Se desejar processar o novo cabeçalho (ou qualquer cabeçalho modificado) em seu pipeline, você deverá definir um outro manipulador de mensagem SOAP em seu pipeline.

Para uma mensagem de saída (uma solicitação em um solicitante de serviço, uma resposta em um provedor de serviços) os manipuladores de mensagem SOAP fornecidos pelo CICS criam uma mensagem SOAP que não contém nenhum cabeçalho. Para incluir um ou mais cabeçalhos na mensagem, você deve gravar um programa de manipulador de cabeçalho para incluir os cabeçalhos. Para assegurar que este manipulador de cabeçalho seja chamado, você deve defini-lo em seu arquivo de configuração de pipeline e especificar <mandatory>true</mandatory>.

Se um manipulador de cabeçalho for chamado na fase de solicitação de um pipeline, ele será chamado novamente na fase de resposta, mesmo se a mensagem que flui na fase de resposta não contiver um cabeçalho correspondente.

A Interface do Programa de Processamento de Cabeçalho

Os manipuladores de mensagem SOAP 1.1 e SOAP 1.2 fornecidos pelo CICS se vinculam aos programas de processamento de cabeçalho usando o canal

DFHHHC-V1. Os contêineres que são transmitidos no canal incluem vários que são específicos para a interface do programa de processamento de cabeçalho e conjuntos de *contêineres de contextos* e *contêineres do usuário* que são acessíveis para todos os programas de processamento de cabeçalho e programas do manipulador de mensagem no pipeline.

O contêiner DFHHEADER é específico para a interface do programa de processamento de cabeçalho. Outros contêineres estão disponíveis em outro lugar em seu pipeline, mas têm utilizações específicas em um programa de processamento de cabeçalho. Os contêineres nesta categoria são DFHWS-XMLNS, DFHWS-BODY e DFHXMLSS-PARSE.

Nota: Embora o serviço da web que usa Axis2 para processar mensagens SOAP possa usar a interface do programa de processamento de cabeçalho, é mais eficiente gravar seus próprios manipuladores Axis2 em Java para processar os cabeçalhos SOAP. Para obter mais informações sobre como criar manipuladores Axis2, consulte Gravando seu Próprio Módulo Axis2

Contêiner DFHHEADER

Quando o programa de processamento de cabeçalho é chamado, DFHHEADER contém o bloco de cabeçalho único que faz com que o programa de processamento de cabeçalho fosse conduzido. Quando o programa do cabeçalho é especificado com `<mandatory>true</mandatory>` ou `<mandatory>1</mandatory>` no arquivo de configuração de pipeline, ele é chamado mesmo quando não há bloco de cabeçalho correspondente na mensagem SOAP. Neste caso, o contêiner DFHHEADER possui um comprimento igual a zero. Este é o caso quando um programa de processamento de cabeçalho é chamado para incluir um bloco de cabeçalho em uma mensagem SOAP que não possui blocos de cabeçalhos.

A mensagem SOAP que o CICS cria não possui cabeçalhos inicialmente. Se desejar incluir cabeçalhos em sua mensagem, você deverá assegurar que pelo menos um programa de processamento de cabeçalho seja chamado, especificando `<mandatory>true</mandatory>` ou `<mandatory>1</mandatory>`.

Quando o programa de cabeçalho retorna, o contêiner DFHHEADER deve conter zero, um ou mais blocos de cabeçalho que o CICS insere na mensagem SOAP no lugar do original:

- É possível retornar o bloco de cabeçalho original inalterado.
- É possível modificar o conteúdo do bloco de cabeçalho.
- É possível anexar um ou mais blocos de cabeçalho novos no bloco original.
- É possível substituir o bloco de cabeçalho original por um ou mais blocos diferentes.
- É possível excluir o bloco de cabeçalho completamente.

Contêiner DFHWS-XMLNS

Quando o programa de processamento do cabeçalho é chamado, DFHWS-XMLNS contém informações sobre namespaces XML que são declarados no envelope SOAP. O programa do cabeçalho pode usar estas informações para executar as tarefas a seguir:

- Resolver nomes qualificados que ele encontra no bloco de cabeçalho
- Construir nomes qualificados nos blocos de cabeçalho novos ou modificados.

As informações de namespace consistem em uma lista de declarações de namespace, que utilizam a notação XML padrão para declarar namespaces. As declarações de namespaces em DFHWS-XMLNS são separadas por espaços. Por exemplo:

```
xmlns:na='http://abc.example.org/schema' xmlns:nx='http://xyz.example.org/schema'
```

É possível incluir declarações de namespace adicionais no envelope SOAP anexando-as no conteúdo de DFHWS-XMLNS. No entanto, namespaces cujo escopo é um bloco de cabeçalho SOAP ou um corpo SOAP são melhor declarados no bloco de cabeçalho ou no corpo, respectivamente. É avisado para não excluir declarações de namespace do contêiner DFHWS-XMLNS em um programa de processamento de cabeçalho, porque os elementos XML que não são visíveis no programa podem depender delas.

Contêiner DFHWS-BODY

Este contêiner contém a seção do corpo do envelope SOAP. O programa de processamento de cabeçalho pode modificar o conteúdo.

Quando o programa de processamento de cabeçalho é chamado, o DFHWS-BODY contém o elemento SOAP <Body>.

Quando o programa de cabeçalho retorna, o contêiner DFHWS-BODY deve novamente conter um <Body> SOAP válido, que o CICS insere na mensagem SOAP no lugar do original:

- É possível retornar o corpo original inalterado.
- É possível modificar o conteúdo do corpo.

Você não deve excluir o corpo SOAP completamente, pois toda mensagem SOAP deve conter um elemento <Body>.

Contêiner DFHXMLSS-PARSE

Ao usar os elementos <cics_soap_1.1_handler> ou <cics_soap_1.2_handler> em sua configuração de pipeline, e o programa do cabeçalho é chamado, DFHXMLSS-PARSE contém os registros XML System Services (XMLSS) para esse cabeçalho. Este contêiner não é criado quando os elementos <cics_soap_1.1_handler_java> ou <cics_soap_1.2_handler_java> são usados.

Contêineres de Controle, Contexto e do Usuário

Assim como os contêineres descritos, a interface transmite os *contêineres de controle*, *contêineres de contextos* e *contêineres do usuário* no canal DFHHHC-V1.

Para obter mais informações sobre estes contêineres, consulte “Contêineres usados no pipeline” na página 148.

Roteamento Dinâmico de Solicitações de Entrada em um Manipulador de Terminal

Quando o manipulador de terminal de um pipeline do provedor de serviços é um dos manipuladores de mensagem SOAP fornecidos pelo CICS, o programa do manipulador de aplicativo de destino especificado no contêiner **DFHWS-APPHANDLER** é, em alguns casos, elegível para roteamento dinâmico. Todo processamento de pipeline antes do programa do manipulador de aplicativo é sempre executado localmente na região CICS que recebeu a mensagem SOAP.

A transação que executa o programa manipulador do aplicativo de destino é elegível para roteamento quando uma das seguintes condições é verdadeira:

- A transação sob a qual o pipeline está processando a mensagem é definida como DYNAMIC ou REMOTE. Essa transação é definida no URIMAP que é utilizado para mapear o URI a partir da mensagem SOAP de entrada.
- Um programa no pipeline mudou o conteúdo do contêiner **DFHWS-USERID** a partir de seu valor inicial.
- Um programa no pipeline mudou o conteúdo do contêiner **DFHWS-TRANID** a partir de seu valor inicial.
- Um cabeçalho SOAP WS-AT existe na mensagem SOAP de entrada.

Em todos os cenários anteriores, um comutador de tarefa ocorre durante o processamento de pipeline. A segunda tarefa é executada sob a transação especificada no contêiner **DFHWS-TRANID**. Esse comutador de tarefa fornece uma oportunidade para que o roteamento dinâmico ocorra, mas somente se o MRO é usado para conectar as regiões CICS juntas. Além disso, a região CICS para a qual você está roteando deve suportar canais e contêineres.

O roteamento ocorrerá somente se a definição de TRANSACTION para a transação nomeada em **DFHWS-TRANID** especificar um dos conjuntos de atributos a seguir:

DYNAMIC(YES)

A transação é roteada utilizando o modelo de roteamento distribuído, no qual o programa de roteamento é especificado no parâmetro de inicialização do sistema **DSRTPGM**.

DYNAMIC(NO) REMOTESYSTEM(sysid)

A transação é roteada para o sistema identificado pelo *sysid*.

Para obter mais informações sobre o roteamento de solicitações de serviço da web, consulte a nota técnica: *Roteamento de serviços da web do CICS no modo de provedor*.

Para os aplicativos implementados com o assistente de serviços da web do CICS, há uma segunda oportunidade para rotear dinamicamente a solicitação, no ponto em que o CICS é vinculado ao programa de usuários. A solicitação é, então, roteada usando o modelo de roteamento dinâmico, no qual o programa de roteamento é especificado no parâmetro de inicialização do sistema **DTRPGM**. A elegibilidade para roteamento é determinada, neste caso, pelas características do programa. Se você estiver usando um canal e contêineres ao vincular-se ao programa, poderá rotear dinamicamente a solicitação somente para regiões CICS que estão na V3.1 ou superior. Se estiver usando uma COMMAREA, essa restrição não se aplica.

Quando uma solicitação tiver sido roteada dinamicamente para uma região de destino, ela não poderá ser roteada dinamicamente do destino para uma terceira região, embora a transação seja definida como ROUTABLE (YES) e DYNAMIC (YES). A transação pode, no entanto, ser roteada estaticamente a partir da região de destino para uma terceira região.

Contêineres usados no pipeline

Um pipeline geralmente consiste de diversos programas manipuladores de mensagens e, quando os manipuladores de mensagens SOAP fornecidos pelo CICS são utilizados, diversos programas de processamento de cabeçalho. O CICS usa

contêineres para transmitir informações para e a partir destes programas. Os programas também usam contêineres para se comunicarem com outros programas no pipeline.

O pipeline do CICS se vincula aos manipuladores de mensagem e aos programas de processamento de cabeçalho usando um canal que possui vários contêineres. Alguns contêineres são opcionais, outros são requeridos por todos os manipuladores de mensagem e outros são usados por alguns manipuladores de mensagem e não por outros.

Antes de um manipulador ser chamado, alguns ou todos os contêineres são preenchidos com informações que o manipulador pode usar para executar seu trabalho. Os contêineres retornados pelo manipulador determinam o processamento subsequente e são transmitidos para manipuladores posteriores no pipeline.

Os contêineres podem ser categorizados destas maneiras:

Contêineres de Controle

Estes contêineres são essenciais para a operação do pipeline. Os manipuladores podem usar os contêineres de controle para modificar a sequência na qual os manipuladores são processados. Os nomes dos contêineres de controle são definidos pelo CICS e iniciam com os caracteres DFH.

Contêineres de Contextos

Estes contêineres contêm informações sobre o ambiente no qual os manipuladores são chamados. O CICS coloca informações nestes contêineres antes de chamar o primeiro manipulador de mensagem mas, em alguns casos, os manipuladores são livres para alterar o conteúdo ou para excluir os contêineres. Mudanças nos contêineres de contextos não afetam diretamente a sequência na qual os manipuladores são chamados. Os nomes dos contêineres de contextos são definidos pelo CICS e iniciam com os caracteres DFH.

Contêineres do Programa de Processamento de Cabeçalho

Estes contêineres contêm informações que são usadas pelos programas de processamento de cabeçalho que são chamados a partir dos manipuladores de mensagem SOAP fornecidos pelo CICS. Para obter mais informações sobre esses containers, consulte A interface do programa de processamento de cabeçalho.

Contêineres de Segurança

Estes contêineres contêm informações que são usadas pela interface do cliente de Confiança e o manipulador de mensagem de segurança para processar tokens de segurança usando um Security Token Service (STS). Os nomes dos contêineres de segurança são definidos pelo CICS e iniciam com os caracteres DFH.

Contêineres Gerados

Estes contêineres contêm os dados da mensagem SOAP, tais como matrizes de variável e sequências longas, que são transmitidas para e a partir do programa de aplicativo para processamento. O CICS cria estes contêineres automaticamente durante o processamento de pipeline e os nomes iniciam com os caracteres DFH.

Contêineres do Usuário

Estes contêineres contêm informações que um manipulador de mensagem precisa transmitir para um outro. O uso de contêineres de usuário é

inteiramente uma questão para os manipuladores de mensagens. É possível escolher seus próprios nomes para estes contêineres, mas você não deve usar nomes que iniciam com DFH.

Contêineres de Controle

Os contêineres de controle são essenciais para a operação do pipeline. Os manipuladores podem usar os contêineres de controle para modificar a sequência na qual os manipuladores são processados.

Contêiner DFHERROR:

DFHERROR é um contêiner de DATATYPE(BIT) que é usado para transmitir informações sobre erros de pipeline para outros manipuladores de mensagem.

Tabela 5. Estrutura do Contêiner DFHERROR.

Nome do campo	Comprimento (bytes)	Conteúdo
PIISNEB-MAJOR-VERSION	1	"1"
PIISNEB-MINOR-VERSION	1	"1"
PIISNEB-ERROR-TYPE	1	Um valor numérico denotando o tipo de erro. Os valores são descritos em Tabela 6.
PIISNEB-ERROR-MODE	1	<p>I O erro ocorreu em um pipeline do provedor</p> <p>A O erro ocorreu em um pipeline do solicitante</p> <p>T O erro ocorreu em um cliente de Confiança</p>
PIISNEB-ABCODE	4	O código de encerramento anormal quando o erro está associado a um encerramento de forma anormal da transação.
PIISNEB-ERROR-CONTAINER1	16	O nome do contêiner quando o erro está associado a um contêiner.
PIISNEB-ERROR-CONTAINER2	16	O nome do segundo contêiner quando o erro está associado a mais de um contêiner.
PIISNEB-ERROR-NODE	8	O nome do programa manipulador no qual o erro ocorreu.

Tabela 6. Valores para o Campo PIISNEB-ERROR-TYPE

Valor de PIISNEB-ERROR-TYPE	Significado
1	O programa manipulador falhou. O código de encerramento anormal está no campo PIISNEB-ABCODE.

Tabela 6. Valores para o Campo PIISNEB-ERROR-TYPE (continuação)

Valor de PIISNEB-ERROR-TYPE	Significado
2	Um contêiner requerido pelo manipulador estava vazio. O nome do contêiner está no campo PIISNEB-ERROR-CONTAINER1.
3	Um contêiner requerido pelo manipulador estava ausente. O nome do contêiner está no campo PIISNEB-ERROR-CONTAINER1.
4	Dois contêineres foram transmitidos ao manipulador quando somente um era esperado. Os nomes dos contêineres estão nos campos PIISNEB-ERROR-CONTAINER1 e PIISNEB-ERROR-CONTAINER2.
5	Uma tentativa de vincular-se ao programa de destino falhou. Se o programa de destino falhou, o código de encerramento anormal está no contêiner PIISNEB-ABCODE.
6	O gerenciador de pipeline falhou em comunicar-se com um servidor remoto devido a um erro no transporte subjacente.
7	O contêiner DFHWS-STSACTION possui um erro. Ele está ausente, corrompido ou contém um valor incorreto.
8	DFHPIRT falhou em iniciar o pipeline.
9	DFHPIRT falhou em colocar uma mensagem em um contêiner.
10	DFHPIRT falhou em obter uma mensagem de um contêiner.
11	Ocorreu um erro não manipulado.

A declaração de COBOL da estrutura do contêiner é esta:

```
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).
```

A tabela a seguir mostra os copybooks de linguagem que mapeiam o contêiner.

Tabela 7. Copybooks que mapeiam o contêiner

Idioma	Copybook
COBOL	DFHPIUCO
PL/I	DFHPIUCL
C e C++	dfhpiuch.h
Assembler	DFHPIUCD

Contêiner DFHFUNCTION:

DFHFUNCTION é um contêiner de DATATYPE(CHAR) que contém uma sequência de 16 caracteres que indica onde em um pipeline um programa está sendo chamado.

A sequência possui um dos valores a seguir. As posições de caractere da extremidade direita são preenchidas com caracteres em branco.

RECEIVE-REQUEST

O manipulador é um manipulador não de terminal em um pipeline do provedor de serviços e está sendo chamado para processar uma mensagem de solicitação de entrada. Na entrada para o manipulador, a mensagem está no contêiner de controle DFHREQUEST.

SEND-RESPONSE

O manipulador é um manipulador não de terminal em um pipeline do provedor de serviços e está sendo chamado para processar uma mensagem de resposta de saída. Na entrada para o manipulador, a mensagem está no contêiner de controle DFHRESPONSE.

SEND-REQUEST

O manipulador está sendo chamado por um pipeline que está enviando uma solicitação; ou seja, em um solicitante de serviço que está processando uma mensagem de saída

RECEIVE-RESPONSE

O manipulador está sendo chamado por um pipeline que está recebendo uma resposta; ou seja, em um solicitante de serviço que está processando uma mensagem de entrada

PROCESS-REQUEST

O manipulador está sendo chamado como o manipulador de terminal de um pipeline do provedor de serviços

NO-RESPONSE

O manipulador está sendo chamado após o processamento de uma solicitação, quando nenhuma resposta deve ser processada.

HANDLER-ERROR

O manipulador está sendo chamado porque um erro foi detectado.

Em um pipeline do provedor de serviços que processa uma solicitação e retorna uma resposta, os valores de DFHFUNCTION que ocorrem são RECEIVE-REQUEST, PROCESS-REQUEST e SEND-RESPONSE. Figura 22 na página 153 mostra a sequência na qual os manipuladores são chamados e os valores de DFHFUNCTION que são transmitidos para cada manipulador.

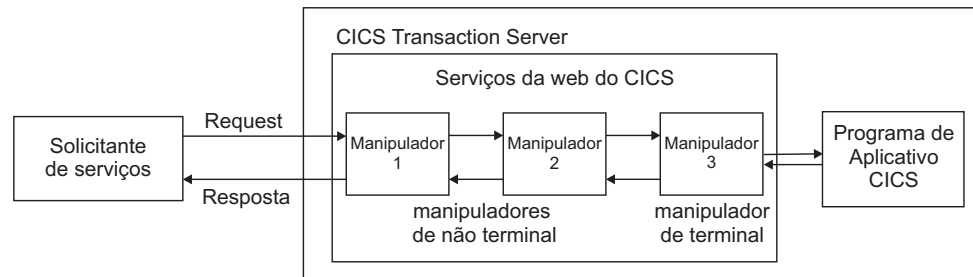


Figura 22. Sequência de Manipuladores em um Pipeline do Provedor de Serviços

Sequência	Rotina de tratamento	DFHFUNCTION
1	Manipulador 1	RECEIVE-REQUEST
2	Manipulador 2	RECEIVE-REQUEST
3	Manipulador 3	PROCESS-REQUEST
4	Manipulador 2	SEND-RESPONSE
5	Manipulador 1	SEND-RESPONSE

Em um pipeline do solicitante de serviço que envia uma solicitação e recebe uma resposta, os valores de DFHFUNCTION que ocorrem são SEND-REQUEST e RECEIVE-RESPONSE. Figura 23 mostra a sequência na qual os manipuladores são chamados e os valores de DFHFUNCTION que são transmitidos a cada manipulador.

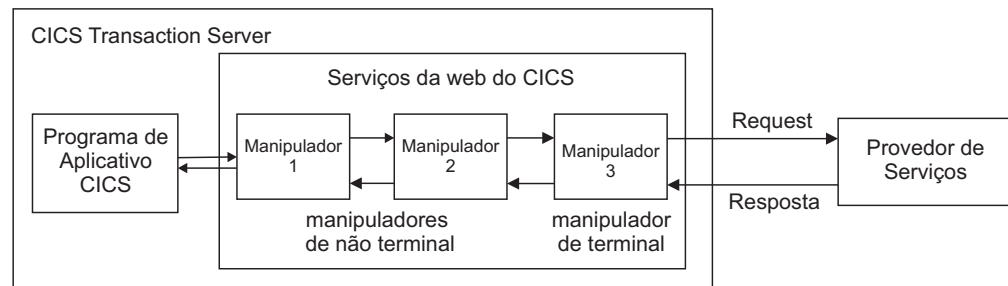


Figura 23. Sequência de Manipuladores em um Pipeline do Solicitante de Serviço

Sequência	Rotina de tratamento	DFHFUNCTION
1	Manipulador 1	SEND-REQUEST
2	Manipulador 2	SEND-REQUEST
3	Manipulador 3	SEND-REQUEST
4	Manipulador 3	RECEIVE-RESPONSE
5	Manipulador 2	RECEIVE-RESPONSE
6	Manipulador 1	RECEIVE-RESPONSE

Os valores de DFHFUNCTION que podem ser encontradas em um determinado manipulador de mensagem dependem de se o pipeline é um provedor ou

solicitante, se o pipeline está na fase de solicitação ou resposta e se o manipulador é um manipulador de terminal ou um manipulador não de terminal. A tabela a seguir resume quando cada valor pode ocorrer:

Valor de DFHFUNCTION	Pipeline do Provedor ou Solicitante	Fase do Pipeline	Manipulador de Terminal ou Não de Terminal
RECEIVE-REQUEST	Provider	Fase de Solicitação	Não de terminal
SEND-RESPONSE	Provider	Fase de Resposta	Não de terminal
SEND-REQUEST	Solicitador	Fase de Solicitação	Não de terminal
RECEIVE-RESPONSE	Solicitador	Fase de Resposta	Não de terminal
PROCESS-REQUEST	Provider	Fase de Solicitação	Terminal
NO-RESPONSE	Ambos	Fase de Resposta	Não de terminal
HANDLER-ERROR	Ambos	Ambos	Ambos

Contêiner DFHHTTPMETHOD:

Este é um contêiner de DATATYPE(CHAR) que está disponível para programas de aplicativos em todos os pipelines do CICS no modo de provedor HTTP.

Este contêiner tem 8 caracteres de comprimento e contém o nome do método HTTP que foi usado na solicitação recebida. Ele não é preenchido se a solicitação não chegou sobre HTTP.

Contêiner DFHHTTPSTATUS:

DFHHTTPSTATUS é um contêiner de DATATYPE(CHAR) que é usado para especificar o código de status de HTTP e o texto de status para uma mensagem produzida na fase de resposta de um pipeline do provedor de serviços.

O conteúdo do contêiner DFHHTTPSTATUS deve ser igual à linha de status inicial de uma mensagem de resposta HTTP, que possui a estrutura a seguir:

HTTP/1.1 nnn tttttttt

HTTP/1.1

A versão e liberação de HTTP.

nnn O código de status HTTP decimal de 3 dígitos para retorno.

tttttttt

O texto de status legível associado ao código de status nnn.

A sequência a seguir é um exemplo do conteúdo:

HTTP/1.1 412 Condição Prévia com Falha

O contêiner DFHHTTPSTATUS é ignorado quando o pipeline usa o transporte do WebSphere MQ.

Se o contêiner contém mais de 45 bytes de dados, o CICS envia 45 bytes e ignora os dados restantes.

Contêiner DFHMEDIATYPE:

DFHMEDIATYPE é um contêiner de DATATYPE(CHAR) que é usado para especificar o tipo de mídia para uma mensagem produzida na fase de resposta de um pipeline do provedor de serviços.

O conteúdo do contêiner DFHMEDIATYPE deve consistir em um tipo e um subtipo separados por um caractere de barra. As sequências a seguir mostram dois exemplos de conteúdo correto para o contêiner DFHMEDIATYPE:

```
texto/plano  
image/svg+xml
```

O contêiner DFHMEDIATYPE é ignorado quando o pipeline usa o transporte do WebSphere MQ.

Contêiner DFHNORESPONSE:

DFHNORESPONSE é um contêiner de DATATYPE(CHAR) que, na fase de solicitação de um pipeline do solicitante de serviço, indica que não espera-se que o provedor de serviços retorne uma resposta.

O conteúdo do contêiner DFHNORESPONSE é indefinido; manipuladores de mensagem que precisam saber se espera-se que o provedor de serviços retorne uma resposta precisam somente determinar se o contêiner está presente ou não:

- Se o contêiner DFHNORESPONSE estiver presente, nenhuma resposta será esperada.
- Se o contêiner DFHNORESPONSE estiver ausente, uma resposta *será* esperada.

Estas informações são fornecidas, inicialmente, pelo aplicativo do solicitante de serviço, com base no protocolo usado com o provedor de serviços. Portanto, é aconselhável que você não exclua este contêiner em um manipulador de mensagem (ou que o crie, se ele não existir), porque fazer isso pode desarranjar o protocolo entre os terminais.

Diferente da fase de solicitação de um pipeline do solicitante de serviço, o uso deste contêiner não é definido.

Contêiner DFHREQUEST:

DFHREQUEST é um contêiner de DATATYPE(CHAR) que contém a mensagem de solicitação que é processada na fase de solicitação de um pipeline.

Se um dos manipuladores de mensagem SOAP fornecidos pelo CICS é configurado no pipeline, o contêiner DFHREQUEST será atualizado para incluir os cabeçalhos de mensagem SOAP no envelope SOAP. Se a mensagem for construída por um manipulador de mensagem SOAP fornecido pelo CICS, e não tiver sido alterada subsequentemente, DFHREQUEST conterá um envelope SOAP completo e todo o seu conteúdo estará na página de códigos UTF-8.

O contêiner DFHREQUEST está presente na solicitação quando um manipulador de mensagem é chamado e o contêiner DFHFUNCTION contém RECEIVE-REQUEST ou SEND-REQUEST.

Nesta situação, o protocolo normal é retornar DFHREQUEST ao pipeline com o mesmo conteúdo ou com conteúdo modificado. O processamento da fase de

solicitação do pipeline continua normalmente, com o próximo programa manipulador de mensagem no pipeline, se houver um.

Como uma alternativa, seu manipulador de mensagem pode excluir o contêiner DFHREQUEST e colocar uma resposta no contêiner DFHRESPONSE. Desta maneira, a sequência normal de processamento é revertida e o processamento continua com a fase de resposta do pipeline.

Contêiner DFHRESPONSE:

DFHRESPONSE é um contêiner de DATATYPE(CHAR) que contém a mensagem de resposta que é processada na fase de resposta de um pipeline. Se a mensagem foi construída por um manipulador de mensagem SOAP fornecido pelo CICS, e não foi alterado subsequentemente, DFHRESPONSE conterá um envelope SOAP completo e todo seu conteúdo na página de códigos UTF-8.

O contêiner DFHRESPONSE está presente quando um manipulador de mensagem é chamado e o contêiner DFHFUNCTION contém SEND-RESPONSE ou RECEIVE-RESPONSE.

Nesta situação, o protocolo normal é retornar DFHRESPONSE ao pipeline com o mesmo conteúdo ou com conteúdo modificado. O processamento de pipeline continua normalmente, com o próximo programa de manipulador de mensagem no pipeline, se houver um.

O contêiner DFHRESPONSE também está presente, com um comprimento igual a zero, quando DFHFUNCTION contém RECEIVE-REQUEST, SEND-REQUEST, PROCESS-REQUEST ou HANDLER-ERROR.

Contêiner DFHWS-CCSID:

DFHWS-CCSID é um contêiner de DATATYPE(BIT) que contém uma palavra inteira (4 bytes) especificando o CCSID dos dados no contêiner de resposta.

O contêiner é válido somente para um pipeline do modo de provedor que utiliza código do CICS para transformar a estrutura de linguagem em XML.

O CCSID deve ser compatível com o CCSID utilizado para gerar o arquivo WSBIND. Se não for, a resposta SOAP que é produzida pode conter caracteres incorretos ou inválidos.

O CCSID não pode ser alterado para ou a partir de 930, 1390, 5026 e 1026. Além disso, o CICS não permite que o CCSID seja alterado para um que seja utilizável como um CCSID do cliente.

Se houver algum problema ao processar o valor no contêiner DFHWS-CCSID, o processamento continuará utilizando o CCSID do arquivo WSBIND.

O contêiner DFHWS-CCSID é verificado somente no retorno de um programa de aplicativo acionado por canal.

Como Contêineres Controlam os Protocolos de Pipeline

Os conteúdos dos contêineres DFHFUNCTION, DFHREQUEST e DFHRESPONSE juntos controlam a protocolos de pipeline.

Durante as duas fases da execução de um pipeline (a fase de solicitação e a fase de resposta) o valor de DFHFUNCIÓN determina quais contêineres de controle são transmitidos para cada manipulador de mensagem:

DFHFUNCIÓN	Contexto	DFHREQUEST	DFHRESPONSE
RECEIVE-REQUEST	Provedor de serviços; fase de solicitação	Presente (comprimento > 0)	Presente (comprimento = 0)
SEND-RESPONSE	Provedor de serviços; fase de resposta	Ausente	Presente (comprimento > 0)
SEND-REQUEST	Solicitante de serviço; fase de solicitação	Presente (comprimento > 0)	Presente (comprimento = 0)
RECEIVE-RESPONSE	Solicitante de serviço; fase de resposta	Ausente	Presente (comprimento > 0)
PROCESS-REQUEST	Provedor de serviços; manipulador de terminal	Presente (comprimento > 0)	Presente (comprimento = 0)
HANDLER-ERROR	Solicitante ou provedor de serviço; qualquer fase	Ausente	Presente (comprimento = 0)
NO-RESPONSE	Solicitante ou provedor de serviço; fase de resposta	Ausente	Ausente

o processamento subsequente é determinado pelo contêineres que seu manipulador de mensagens transmite de volta para o pipeline:

Durante a fase de solicitação

- Seu manipulador de mensagem pode retornar o contêiner DFHREQUEST. O processamento continua na fase de solicitação com o próximo manipulador. O comprimento dos dados no contêiner não deve ser zero.
- Seu manipulador de mensagem pode retornar o contêiner DFHRESPONSE. O processamento alterna para a fase de resposta e o mesmo manipulador é chamado com DFHFUNCIÓN configurado como SEND-RESPONSE em um provedor de serviços e com RECEIVE-RESPONSE em um solicitante de serviço. O comprimento dos dados no contêiner não deve ser zero.
- Seu manipulador de mensagem pode não retornar nenhum contêiner. O processamento alterna para a fase de resposta e o mesmo manipulador é chamado com DFHFUNCIÓN configurado como NO-RESPONSE.

No manipulador terminal (somente provedor de serviços)

- Seu manipulador de mensagem pode retornar o contêiner DFHRESPONSE. O processamento alterna para a fase de resposta e o manipulador anterior é chamado com um novo valor igual a DFHFUNCIÓN (SEND-RESPONSE). O comprimento dos dados no contêiner não deve ser zero.

- Seu manipulador de mensagem pode não retornar nenhum contêiner. O processamento alterna para a fase de resposta e o manipulador anterior é chamado com um novo valor igual a DFHFUNCTION (NO-RESPONSE).

Durante a fase de resposta

- Seu manipulador de mensagem pode retornar o contêiner DFHRESPONSE. O processamento continua na fase de resposta e o próximo manipulador é chamado. O comprimento dos dados no contêiner não deve ser zero.
- Seu manipulador de mensagem pode não retornar nenhum contêiner. O processamento continua na fase de resposta e o próximo manipulador na sequência é chamado com um novo valor igual a DFHFUNCTION (NO-RESPONSE).

Importante: Durante a fase de solicitação, seu manipulador de mensagem pode retornar DFHREQUEST ou DFHRESPONSE, mas não ambos. Como ambos os contêineres estão presentes quando seu manipulador de mensagem é chamado, você deve excluir um deles.

Esta tabela mostra a ação executada pelo pipeline para todos os valores de DFHFUNCTION e todas as combinações de DFHREQUEST e DFHRESPONSE retornadas por cada manipulador de mensagens.

DFHFUNCTION	Contexto	DFHREQUEST	DFHRESPONSE	Ação
RECEIVE-REQUEST	Provedor de serviços; fase de solicitação	Presente (comprimento > 0)	Presente	(erro)
RECEIVE-REQUEST	Provedor de serviços; fase de solicitação	Presente (comprimento > 0)	Ausente	Chame o próximo manipulador com a função RECEIVE-REQUEST
RECEIVE-REQUEST	Provedor de serviços; fase de solicitação	Presente (comprimento = 0)	Não aplicável	(erro)
RECEIVE-REQUEST	Provedor de serviços; fase de solicitação	Ausente	Presente (comprimento > 0)	Altere para a fase de resposta e chame o mesmo manipulador com a função SEND-RESPONSE
RECEIVE-REQUEST	Provedor de serviços; fase de solicitação	Ausente	Presente (comprimento = 0)	(erro)
RECEIVE-REQUEST	Provedor de serviços; fase de solicitação	Ausente	Ausente	Chame o mesmo manipulador com a função NO-RESPONSE
SEND-RESPONSE	Provedor de serviços; fase de resposta	Não aplicável	Presente (comprimento > 0)	Chame o manipulador anterior com a função SEND-RESPONSE
SEND-RESPONSE	Provedor de serviços; fase de resposta	Não aplicável	Presente (comprimento = 0)	(erro)
SEND-RESPONSE	Provedor de serviços; fase de resposta	Não aplicável	Ausente	Chame o mesmo manipulador com a função NO-RESPONSE
SEND-REQUEST	Solicitante de serviço; fase de solicitação	Presente (comprimento > 0)	Presente (comprimento ≥ 0)	(erro)
SEND-REQUEST	Solicitante de serviço; fase de solicitação	Presente (comprimento > 0)	Ausente	Chame o próximo manipulador com a função SEND-REQUEST

DFHFUNCTION	Contexto	DFHREQUEST	DFHRESPONSE	Ação
SEND-REQUEST	Solicitante de serviço; fase de solicitação	Presente (comprimento = 0)	Não aplicável	(erro)
SEND-REQUEST	Solicitante de serviço; fase de solicitação	Ausente	Presente (comprimento > 0)	Alterne para a fase de resposta e chame o manipulador anterior com a função RECEIVE-RESPONSE
SEND-REQUEST	Solicitante de serviço; fase de solicitação	Ausente	Presente (comprimento = 0)	(erro)
SEND-REQUEST	Solicitante de serviço; fase de solicitação	Ausente	Ausente	Chame o mesmo manipulador com a função NO-RESPONSE
RECEIVE-RESPONSE	Solicitante de serviço; fase de resposta	Não aplicável	Presente (comprimento > 0)	Chame o manipulador anterior com a função RECEIVE-RESPONSE
RECEIVE-RESPONSE	Solicitante de serviço; fase de resposta	Não aplicável	Presente (comprimento = 0)	(erro)
RECEIVE-RESPONSE	Solicitante de serviço; fase de resposta	Não aplicável	Ausente	Chame o mesmo manipulador com a função NO-RESPONSE
PROCESS-REQUEST	Provedor de serviços; manipulador de terminal	Não aplicável	Presente (comprimento > 0)	Chame o manipulador anterior com a função RECEIVE-RESPONSE
PROCESS-REQUEST	Provedor de serviços; manipulador de terminal	Não aplicável	Presente (comprimento = 0)	(erro)
PROCESS-REQUEST	Provedor de serviços; manipulador de terminal	Não aplicável	Ausente	Chame o mesmo manipulador com a função NO-RESPONSE
HANDLER-ERROR	Solicitante ou provedor de serviço; qualquer fase	Não aplicável	Presente (comprimento > 0)	Chame o manipulador anterior com a função SEND-RESPONSE ou a função RECEIVE-RESPONSE
HANDLER-ERROR	Solicitante ou provedor de serviço; qualquer fase	Não aplicável	Presente (comprimento = 0)	(erro)
HANDLER-ERROR	Solicitante ou provedor de serviço; qualquer fase	Não aplicável	Ausente	Chame o mesmo manipulador com a função NO-RESPONSE

Contêineres de Contextos

Em algumas situações, programas de manipulador de mensagem gravados pelo usuário, e programas de processamento de cabeçalho, precisam de informações sobre o contexto no qual eles são chamados. O CICS fornece estas informações em um conjunto de *contêineres de contextos*, que são transmitidos para os programas.

O CICS inicializa o conteúdo de cada contêiner de contextos mas, em alguns casos, é possível alterar o conteúdo em seus programas de manipulador de mensagem e no programa de processamento de cabeçalho. Por exemplo, em um pipeline do provedor de serviços no qual o manipulador de terminal é um dos manipuladores

SOAP fornecidos pelo CICS, é possível alterar o ID do usuário e o ID da transação do programa de aplicativo de destino modificando o conteúdo dos contêineres de contextos apropriados.

Algumas das informações fornecidas nos contêineres se aplicam somente a um provedor de serviços ou somente a um solicitante de serviço e, portanto, alguns dos contêineres de contextos não estão disponíveis em ambos.

Contêiner DFH-EXIT-HEADER1:

DFH-EXIT-HEADER1 é um contêiner de DATATYPE(CHAR). Ele contém um ou mais cabeçalhos SOAP que são incluídos em uma resposta a partir de um aplicativo do provedor de serviço da web no CICS.

Programas que executam a saída de usuário global XWSPRRWO podem incluir um cabeçalho em uma resposta SOAP. O cabeçalho deve ser um SOAP válido e os namespaces devem ser autocontidos no cabeçalho XML. Um programa que coloca dados neste contêiner deve verificar sua presença e incluir o novo cabeçalho no final dos dados. Seguindo esta melhor prática, diversos programas podem ser conduzidos no mesmo ponto de saída, se necessário.

Contêiner DFH-HANDLERPLIST:

DFH-HANDLERPLIST é um contêiner de DATATYPE(CHAR) que é inicializado com o conteúdo do elemento `<handler_parameter_list>` apropriado do arquivo de configuração de pipeline.

Se você não tiver especificado uma lista de parâmetros do manipulador no arquivo de configuração de pipeline, o contêiner estará vazio; ou seja, ele terá um comprimento igual a zero.

Não é possível alterar o conteúdo deste contêiner.

Contêiner DFH-SERVICEPLIST:

DFH-SERVICEPLIST é um contêiner de DATATYPE(CHAR) que contém o conteúdo do elemento `<service_parameter_list>` do arquivo de configuração de pipeline.

Se você não tiver especificado uma lista de parâmetros de serviço no arquivo de configuração de pipeline, o contêiner estará vazio; ou seja, ele terá um comprimento zero.

Não é possível alterar o conteúdo deste contêiner.

Contêiner DFHWS-APPHANDLER:

DFHWS-APPHANDLER é um contêiner de DATATYPE(CHAR) que, em um pipeline do provedor de serviços, é inicializado com o conteúdo do elemento `<apphandler>` do arquivo de configuração de pipeline.

No manipulador de terminal de um pipeline que contém o elemento `<apphandler>`, os manipuladores SOAP fornecidos obtêm o nome do programa aplicativo de destino a partir deste contêiner.

É possível alterar o conteúdo desse contêiner em seus manipuladores de mensagem ou programas de processamento de cabeçalho.

O CICS não fornecer este contêiner em um pipeline do solicitante de serviço.

Conceitos relacionados:

“Manipuladores de Aplicativo” na página 90

Um manipulador de aplicativo é um programa CICS que o manipulador de terminal de um pipeline do provedor de serviços SOAP vincula ao tempo de execução.

Contêiner DFHWS-APPHANCLAS:

DFHWS-APPHANCLAS é um contêiner de DATATYPE(CHAR) que, em um pipeline do provedor de serviços, é inicializado com o conteúdo do elemento `<apphandler_class>` do arquivo de configuração de pipeline.

No manipulador de terminal de um pipeline baseado em Java, os manipuladores SOAP fornecidos, `<cics_soap_1.1_handler_java>` e `<cics_soap_1.2_handler_java>`, obtêm o nome do programa de aplicativo de destino deste contêiner.

O CICS não fornecer este contêiner em um pipeline do solicitante de serviço.

Conceitos relacionados:

“Manipuladores de Aplicativo” na página 90

Um manipulador de aplicativo é um programa CICS que o manipulador de terminal de um pipeline do provedor de serviços SOAP vincula ao tempo de execução.

Referências relacionadas:

“O Elemento `<apphandler_class>`” na página 91

Especifica que o manipulador do terminal do pipeline vincula-se a um manipulador de aplicativo Axis2.

Contêiner DFHWS-DATA:

DFHWS-DATA é um contêiner de DATATYPE(BIT) que é usado em aplicativos do solicitante de serviço e, opcionalmente, em aplicativos do provedor de serviços que são implementados com o assistente de serviços da web do CICS. Ele contém a estrutura de dados de nível superior que é mapeada para e a partir de uma solicitação SOAP.

Em aplicativos do solicitante de serviço, o contêiner DFHWS-DATA deve estar presente quando o programa do solicitante de serviço emite um comando **EXEC CICS INVOKE SERVICE**. Quando o comando é emitido, o CICS converte a estrutura de dados que está no contêiner em uma solicitação SOAP. Quando a resposta de SOAP é recebida, o CICS a converte em uma outra estrutura de dados que é retornada para o aplicativo no mesmo contêiner.

Em aplicativos do provedor de serviços, o contêiner DFHWS-DATA é usado por padrão quando você não especifica o parâmetro **CONTID** nas tarefas em lote DFHLS2WS ou DFHWS2LS. O CICS converte a mensagem de solicitação SOAP na estrutura de dados que é transmitida ao aplicativo no contêiner DFHWS-DATA. A resposta é, então, salva no mesmo contêiner e o CICS converte a estrutura de dados em uma mensagem de resposta SOAP.

Contêiner DFHWS-FAULT:

O DFHWS-FAULT é um contêiner de DATATYPE(BIT) que contém informações sobre o tipo de falha de SOAP que o CICS gera.

O contêiner contém uma palavra inteira binária que indica o tipo de falha que pode ser usado em processamento adicional para uma resposta de serviço da web:

1. A falha de SOAP mais recente foi para uma falha do CICS (por exemplo, encerramento de forma anormal do CICS ou do usuário).
2. A falha de SOAP mais recente foi para uma falha do aplicativo. O contêiner é excluído quando você emite o comando EXEC CICS SOAPFAULT DELETE. Se uma segunda ou nova falha de SOAP for criada, o CICS atualizará o novo contêiner apropriadamente.

Não é possível alterar o conteúdo deste contêiner.

Contêiner DFHWS-LOCATION:

DFHWS-LOCATION é um contêiner do DATATYPE(CHAR) que contém um cabeçalho Local fornecido quando a resposta do HTTP tiver sido 302, 303 ou 307.

Contêiner DFHWS-MEP:

DFHWS-MEP é um contêiner de DATATYPE(BIT) que contém um valor representativo para o padrão de troca de mensagens (MEP) de uma mensagem SOAP de entrada ou saída. Este valor tem um byte de comprimento.

O CICS suporta quatro padrões de troca de mensagens para solicitantes de serviço e provedores de serviços. O padrão de troca de mensagens é definido no documento WSDL 2.0 para o serviço da web e determina se o CICS responde como o provedor e se o CICS espera uma resposta de um provedor externo. No modo solicitante, o tempo que o CICS espera por uma resposta é configurado usando o recurso PIPELINE.

Se você usou o assistente de serviços da web do CICS para implementar seu aplicativo, este contêiner será preenchido pelo CICS:

- Em um pipeline do provedor de serviços, este contêiner é preenchido pelo manipulador de aplicativo DFHPITP quando ele recebe a mensagem de entrada do manipulador de terminal.
- Em um pipeline do solicitante de serviço, este contêiner é preenchido quando o aplicativo usa o comando **INVOKE SERVICE**.

Se o aplicativo usar o canal DFHPIRT para iniciar o pipeline, o aplicativo preencherá este contêiner. Se o contêiner não estiver presente ou não tiver valor, o CICS assumirá que a solicitação está usando o MEP In-Out ou In-Only, dependendo de se o contêiner DFHNORESPONSE está presente no canal.

Este contêiner é preenchido pelo programa manipulador de aplicativo fornecido, DFHPITP. Se você usar um manipulador de aplicativo diferente, este contêiner não estará disponível para uso.

Tabela 8. Valores que Podem Aparecer no Contêiner DFHWS-MEP

Mínimo	MEP	URI
1	In-Only	http://www.w3.org/ns/wSDL/in-only

Tabela 8. Valores que Podem Aparecer no Contêiner DFHWS-MEP (continuação)

Mínimo	MEP	URI
2	In-Out	http://www.w3.org/ns/wsdl/in-out
4	Robust-In-Only	http://www.w3.org/ns/wsdl/robust-in-only
8	In-Optional-Out	http://www.w3.org/ns/wsdl/in-opt-out

Contêiner DFHWS-OPERATION:

DFHWS-OPERATION é um contêiner de DATATYPE(CHAR) que geralmente é usado em um aplicativo do provedor de serviços implementado com o assistente de serviços da web do CICS. Ele contém o nome da operação que é especificada em uma solicitação de SOAP.

Em um provedor de serviços, o contêiner fornece o nome da operação para a qual o aplicativo está sendo chamado. Ele é preenchido quando um manipulador de mensagem SOAP fornecido passa o controle ao programa do aplicativo de destino e fica visível somente quando o programa de destino é chamado com uma interface do canal.

Em um pipeline do solicitante de serviço, o contêiner contém o nome especificado na opção OPERATION do comando **EXEC CICS INVOKE SERVICE**. O contêiner não está disponível para o aplicativo que emite o comando.

Este contêiner é preenchido pelo programa manipulador de aplicativo fornecido, DFHPITP. Se você usar um manipulador de aplicativo diferente, este contêiner não estará disponível para uso.

Contêiner DFHWS-PIPELINE:

DFHWS-PIPELINE é um contêiner de DATATYPE(CHAR) que contém o nome do PIPELINE no qual o programa está sendo executado.

Não é possível alterar o conteúdo deste contêiner.

Contêiner DFHWS-RESPWAIT:

DFHWS-RESPWAIT é um contêiner de DATATYPE(BIT) que contém um número binário de palavra inteira não assinado para representar o tempo limite em segundos que se aplica à solicitação de serviço da web de saída e mensagens de resposta.

O valor deste contêiner é fornecido pelo atributo RESPWAIT da definição PIPELINE e é configurado pelo CICS quando o comando INVOKE SERVICE é emitido. Qualquer valor configurado neste contêiner pelo aplicativo de usuário antes do comando INVOKE SERVICE ser emitido será ignorado.

Um programa manipulador de mensagem que é chamado durante o processamento de pipeline pode sobrescrever o valor do contêiner DFHWS-RESPWAIT. Se isso for feito, o valor atualizado será usado somente se a definição PIPELINE possuir um atributo RESPWAIT que não tenha sido configurado para DEFT ou deixado em branco. Se a definição PIPELINE tiver o atributo RESPWAIT configurado como DEFT ou deixado em branco, o valor de tempo limite padrão do protocolo de transporte sempre será usado, independentemente do valor no contêiner DFHWS-RESPWAIT.

Este contêiner é usado somente nos pipelines do modo do solicitante.

Contêiner DFHWS-SOAPLEVEL:

DFHWS-SOAPLEVEL é um contêiner de DATATYPE(BIT) que contém informações sobre o nível de SOAP usado na mensagem que está sendo processada.

O contêiner contém uma palavra inteira binária que indica o nível de SOAP que é usado para uma solicitação ou resposta de serviço da web:

- 1 A solicitação ou resposta é uma mensagem SOAP 1.1.
- 2 A solicitação ou resposta é uma mensagem SOAP 1.2.
- 10 A solicitação ou resposta não é uma mensagem SOAP.

Não é possível alterar o conteúdo deste contêiner.

Contêiner DFHWS-TRANID:

DFHWS-TRANID é um contêiner de DATATYPE(CHAR) que é inicializado com o ID da transação da tarefa na qual o pipeline está em execução.

Se você alterar o conteúdo deste contêiner em um pipeline do provedor de serviços no qual o manipulador de terminal é um dos manipuladores fornecidos pelo CICS (e você o fizer antes do controle ser transmitido ao programa do aplicativo de destino), o aplicativo de destino será executado em uma nova tarefa com o novo ID de transação.

Novas tarefas não podem ser iniciadas quando o manipulador de terminal e o manipulador de aplicativo de um pipeline são executados no mesmo servidor JVM. Por esse motivo, se você implementar os aplicativos JAX-WS Axis2 no CICS, o DFHWS-TRANID não poderá ser usado para mudar o ID do usuário.

Contêiner DFHWS-URI:

O DFHWS-URI é um contêiner de DATATYPE(CHAR) que contém o URI do serviço.

Em um pipeline do provedor de serviços o, CICS extrai o URI relativo da mensagem recebida e o coloca no contêiner DFHWS-URI.

Por exemplo, se o URI dos serviços da web for `http://example.com/location/address` ou `jms://queue?destination=INPUT.QUEUE&targetService=/location/address`, o URI relativo será `/location/address`.

Se você estiver usando o Web Services Addressing em seu pipeline do solicitante, esse contêiner será criado e atualizado na seguinte ordem:

- 1. Quando o comando `INVOKE SERVICE` é executado, ele cria o contêiner DFHWS-URI e o inicia com o valor do endereço do terminal de serviço WSDL. Se o comando `WSACONTEXT BUILD API` foi usado para criar um contexto de endereçamento, você não deve especificar os parâmetros `URI` ou `URIMAP` no comando `INVOKE SERVICE`.
- 2. Quando o manipulador de endereçamento de serviços da web (DFHWSADH) é executado, se existe uma `EPR <wsa:To>` no contexto do endereçamento com um URI não anônimo, o URI no contêiner DFHWS-URI é sobrescrito pelo valor da `EPR <wsa:To>`. O URI anônimo é ignorado.

A mensagem SOAP é enviada ao serviço definido pelo URI em DFHWS-URI.

Em um pipeline do solicitante de serviço, o CICS coloca o URI que é especificado no comando **INVOKE SERVICE** ou, se ausente, o URI da ligação de serviço da web, no contêiner DFHWS-URI. É possível substituir este URI usando um manipulador de mensagem no pipeline.

Um serviço pode usar um URI HTTP, HTTPS, JMS ou WebSphere MQ para serviços externos. Um serviço também pode usar um URI do CICS para um serviço que é fornecido por um outro aplicativo CICS:

URI	Sequência de consultas	Descrição
<code>cics://PROGRAM/program</code>	<code>?options</code>	O manipulador de transporte CICS usa um comando EXEC CICS LINK PROGRAM para vincular-se ao programa, especificado transmitindo o canal e os contêineres atuais. Não ocorre nenhuma transformação de dados nos dados do aplicativo.
<code>cics://SERVICE/service</code>	<code>?targetServiceUri=targetServiceUri &options</code>	O manipulador de transporte CICS usa o caminho do serviço, expresso como o <code>targetServiceUri</code> , para corresponder um recurso URIMAP para executar a solicitação por meio de um pipeline do provedor. Você deve especificar um valor para o parâmetro targetServiceUri se usar este tipo de URI.
<code>cics://PIPELINE/pipeline</code>	<code>?targetServiceUri=targetServiceUri</code>	O manipulador de transporte CICS inicia um outro pipeline do solicitante de serviço.

É possível incluir parâmetros em cada tipo de URI do CICS usando o formato *parameter=value*, no qual cada parâmetro é separado por um e comercial. As regras a seguir se aplicam ao URI do CICS:

- O primeiro parâmetro na sequência de consultas deve ser prefixado com um ponto de interrogação. Não é possível usar um ponto de interrogação antes deste ponto no URI.
- Para incluir um e comercial em um valor de parâmetro, você deve escapar o caractere. Para obter mais informações, consulte a seção de exemplo no final deste tópico.
- O CICS altera quaisquer valores em minúscula para *program* e *pipeline* para maiúsculas.

Os parâmetros na sequência de consultas determinam como o CICS processa a solicitação no final do pipeline do solicitante:

maxCommareaLength=value

Especifique o tamanho máximo da COMMAREA em bytes, que é necessário para o programa do aplicativo de destino. O valor não deve exceder 32763. Se este parâmetro estiver presente na sequência de consultas, o CICS vinculará ao programa especificado usando uma COMMAREA. Se este parâmetro não estiver presente na sequência de consultas, o CICS vinculará ao programa especificado usando um canal.

Este parâmetro não faz distinção entre maiúsculas e minúsculas e é válido somente para o URI `cics://PROGRAM`.

newTask=yes|no

Especifique se o manipulador de transporte executará a solicitação como uma nova tarefa.

Esse parâmetro não faz distinção entre maiúsculas e minúsculas.

`cics://PROGRAM/testapp?newTask=yes` e `cics://PROGRAM/testapp?NEWTASK=Yes` são iguais.

targetServiceUri=uri

Especifique o caminho do serviço a ser chamado. Em um tipo de destino `SERVICE`, o manipulador de transporte usa o valor com `host=localhost` para localizar o recurso `URIMAP` para iniciar um pipeline do provedor de serviços. Em um tipo de destino `PIPELINE`, o manipulador de transporte usa o valor para iniciar um outro pipeline do solicitante.

Esse parâmetro faz distinção entre maiúsculas e minúsculas.

transid=char(4)

Especifique uma transação sob a qual a solicitação será executada. O manipulador de transporte inicia um fluxo de solicitação usando o ID de transação especificado.

Esse parâmetro faz distinção entre maiúsculas e minúsculas.

userid=char(8)

Especifique um ID do usuário sob o qual a solicitação será executada. O manipulador de transporte inicia um fluxo de solicitação usando o ID do usuário especificado.

Esse parâmetro não faz distinção entre maiúsculas e minúsculas.

Tipo de Destino	Parâmetros no URI	
PROGRAM	ID do usuário	Opcional
PROGRAM	transid	Opcional
PROGRAM	maxCommareaLength	Opcional
PROGRAM	newTask	Opcional. Deve ter sim ou não especificado se você especificar userid ou transid .
PROGRAM	targetServiceUri	Não-suportado
SERVICE	ID do usuário	Opcional
SERVICE	transid	Opcional
SERVICE	maxCommareaLength	Não-suportado
SERVICE	newTask	Opcional. Deve ter sim ou não especificado se você especificar userid ou transid .
SERVICE	targetServiceUri	Exigido
PIPELINE	ID do usuário	Não-suportado
PIPELINE	transid	Não-suportado
PIPELINE	maxCommareaLength	Não-suportado
PIPELINE	newTask	Não-suportado
PIPELINE	targetServiceUri	Exigido

Exemplos de URIs do CICS

Neste primeiro exemplo, o contêiner DFHWS-URI possui o URI a seguir no momento que ele atinge o final do pipeline:

```
cics://PROGRAM/testapp?newTask=yes&userid=user1
```

O manipulador de transporte vincula ao programa CICS chamado testapp, transmitindo o canal e os contêineres. Não ocorre nenhuma transformação de dados, portanto, o programa de destino deve estar apto a processar o conteúdo dos contêineres no canal atual. O CICS vincula-se ao programa sob uma nova unidade de trabalho e um ID do usuário diferente de user1.

Neste segundo exemplo, o contêiner DFHWS-URI possui o URI a seguir no momento em que ele atinge o final do pipeline:

```
cics://SERVICE/getStockQuote?targetServiceUri=/stock/getQuote&newTask=yes&userid=user2
```

O manipulador de transporte substitui o URI no contêiner DFHWS-URI pelo valor /stock/getQuote, localiza o URIMAP usando o caminho no parâmetro **targetServiceUri** para resolver o URI e inicia o pipeline do provedor sob uma nova tarefa e ID do usuário diferente.

Neste terceiro exemplo, o contêiner DFHWS-URI possui o URI a seguir no momento em que ele atinge o final do pipeline:

```
cics://PIPELINE/reqpipeA?targetServiceUri=cics://PROGRAM/testapp?newTask=yes%26userid=user1
```

O manipulador de transporte substitui o URI no contêiner DFHWS-URI pelo valor cics://PROGRAM/testapp?newTask=yes&userid=user1 e iniciar o pipeline do solicitante chamado reqpipeA, transmitindo o canal e os contêineres atuais. Os caracteres %26 escapam o e comercial, portanto, o manipulador de transporte coloca o URI inteiro no contêiner DFHWS-URI.

Contêiner DFHWS-URI-RESID:

Este é um contêiner de DATATYPE(CHAR) que é disponível apenas para aplicativos conectados por um pipeline JSON.

Esse contêiner mantém uma cópia simplificada do caminho de URI (um IDentificador de REcurso), no qual o fragmento de URI de caminho que foi usado para a correspondência URIMAP foi removido. Por exemplo,

Se o URIMAP que correspondeu à solicitação recebida tivesse um PATH igual a:
/JSONServices/CustomDetails/*

e o URI recebido do cliente fosse:

```
http://www.example.org:10000/JSONServices/CustomDetails/customerNumber/13388?action=query
```

o conteúdo de DFHWS-URI-RESID seria:

```
customerNumber/13388
```

Aplicativos RESTful JSON poderão usar este contêiner para ajudar a identificar o ID do recurso (ou chave primária) para recursos RESTful que são correspondidos usando um URIMAP com caracteres curinga. Isso deve ser significativamente mais simples do que a análise pelo conteúdo de DFHWS-URI.

Nota: Se o atributo PATH do URIMAP correspondente não for curinga (ou seja, ele contém o caminho completo para o URI), o conteúdo desse contêiner estará vazio.

Nota: O atributo PATH do URIMAP correspondente pode conter um fragmento de sequência de consultas opcional. Em caso afirmativo, o fragmento de sequência de consultas é ignorado ao construir este contêiner.

Contêiner DFHWS-URI-QUERY:

Este é um contêiner de DATATYPE(CHAR), que está disponível para programas de aplicativos em todos pipelines do CICS no modo de provedor HTTP.

Esse contêiner mantém o fragmento de sequência de consultas do URI. Por exemplo,

Se o URI recebido do cliente fosse:

`http://www.example.org:10000/JSONServices/CustomerDetails/customerNumber/13388?action=query&page=1`

o conteúdo de DFHWS-URI-QUERY seria:

`action=query&page=1`

Os aplicativos podem analisar pelo conteúdo desse Contêiner para identificar parâmetros **name=value** individuais do URI.

Nota: Se o URI recebido não incluiu uma sequência de consultas, este Contêiner não estará presente no Canal.

Contêiner DFHWS-URIMAPPATH:

Este é um contêiner de DATATYPE(CHAR) que mantém uma cópia dos dados PATH do URIMAP que foi utilizada para corresponder ao URI de entrada.

Qualquer aplicativo conectado ao pipeline pode fazer uso deste Contêiner para entender mais sobre como ele foi conectado.

Contêiner DFHWS-USERID:

DFHWS-USERID é um contêiner de DATATYPE(CHAR) que é inicializado com o ID do usuário da tarefa na qual o pipeline está em execução.

Se você alterar o conteúdo deste contêiner em um pipeline do provedor de serviços no qual o manipulador de terminal é um dos manipuladores SOAP fornecidos pelo CICS (e você o fizer antes do controle ser transmitido ao programa de aplicativo de destino), o aplicativo de destino será executado em uma nova tarefa que está associada ao novo ID do usuário. A menos que você altere o conteúdo do contêiner DFHWS-TRANID, a nova tarefa possui o mesmo ID de transação que a tarefa na qual o pipeline está em execução.

Novas tarefas não podem ser iniciadas quando o manipulador de terminal e o manipulador de aplicativo de um pipeline são executados no mesmo servidor JVM. Por esse motivo, se você implementar aplicativos JAX-WS Axis2 no CICS, DFHWS-USERID não poderá ser usado para mudar o ID do usuário.

Contêiner DFHWS-WEBSERVICE:

DFHWS-WEBSERVICE é um contêiner de DATATYPE(CHAR) que é usado somente em um pipeline do provedor de serviços. Ele contém o nome do serviço da web que especifica o ambiente de execução quando o aplicativo de destino foi implementado usando o assistente de serviços da web.

O CICS não fornecer este contêiner em um pipeline do solicitante de serviço.

Contêiner DFHWS-CID-DOMAIN:

DFHWS-CID-DOMAIN é um caractere de DATATYPE(CHAR). Ele contém o nome de domínio que é usado para gerar valores de ID de conteúdo para referenciar anexos binários.

O valor do nome de domínio é `cicsts` por padrão. É possível substituir o valor especificando o elemento `<mime_options>` no arquivo de configuração de pipeline.

Não é possível alterar o conteúdo deste contêiner.

Contêiner DFHWS-MTOM-IN:

DFHWS-MTOM-IN é um contêiner de DATATYPE(BIT) que contém informações sobre as opções especificadas para o elemento `<cics_mtom_handler>` do arquivo de configuração de pipeline e informações sobre o formato da mensagem que foi recebido no pipeline.

Ele contém as informações para processar uma mensagem MTOM de entrada no pipeline. A mensagem de entrada pode ser uma mensagem de solicitação de um solicitante de serviço da web ou uma mensagem de resposta de um provedor de serviço da web.

Se você não especificar um manipulador `<cics_mtom_handler>` no arquivo de configuração de pipeline, ou se uma mensagem SOAP for recebida em vez de uma mensagem MTOM, este contêiner não será criado.

Se a segurança de serviços da web estiver configurada no pipeline, ou se a validação for ativada para um serviço da web, o conteúdo do campo `XOP_MODE` em DFHWS-MTOM-IN poderá ser substituído pelo CICS quando o contêiner for criado. Por exemplo, se configurar o pipeline para processar o conteúdo de mensagens MTOM no modo direto e, em seguida, ativar a validação para o serviço da web, o CICS substituirá o valor definido no arquivo de configuração de pipeline e configurará o processamento de XOP para executar no modo de compatibilidade. O CICS executa a substituição devido às restrições no suporte para processar documentos XOP e anexos binários no pipeline.

Não é possível alterar o conteúdo deste contêiner.

Tabela 9. Estrutura do Contêiner DFHWS-MTOM-IN

Nome do campo	Comprimento (bytes)	Conteúdo
MTOM_STATUS	4	Contém o valor "1", indicando que a mensagem recebida pelo CICS está no formato MTOM.
MTOMNOXOP_STATUS	4	Contém um dos valores a seguir: 0 A mensagem MTOM contém anexos binários. 1 A mensagem MTOM não contém anexos binários.

Tabela 9. Estrutura do Contêiner DFHWS-MTOM-IN (continuação)

Nome do campo	Comprimento (bytes)	Conteúdo
XOP_MODE	4	<p>Contém um dos valores a seguir:</p> <p>0 Não ocorre nenhum processamento de XOP.</p> <p>1 O processamento de XOP ocorre no modo de compatibilidade.</p> <p>2 O processamento de XOP ocorre no modo direto.</p>

Contêiner DFHWS-MTOM-OUT:

DFHWS-MTOM-OUT é um contêiner de DATATYPE(BIT) que contém informações sobre as opções especificadas para o elemento <cics_mtom_handler> do arquivo de configuração de pipeline.

Ele contém as informações para processar uma mensagem MTOM de saída no pipeline, se ela é uma mensagem de resposta para um solicitante de serviço da web ou uma mensagem de solicitação para um provedor de serviço da web.

Se você não especificar um elemento <cics_mtom_handler> no arquivo de configuração de pipeline ou se o elemento <mtom_options> no arquivo de configuração de pipeline tiver o atributo send_mtom="no", este contêiner não será criado.

No modo de provedor, este contêiner é criado ao mesmo tempo que o contêiner DFHWS-MTOM-IN. Se o elemento <mtom_options> no arquivo de configuração de pipeline tiver o atributo send_mtom="same", o campo MTOM_STATUS será configurado para indicar se o solicitante de serviço da web deseja uma mensagem de resposta MTOM ou SOAP.

Se a segurança de serviços da web estiver configurada no pipeline, ou se a validação for ativada para um serviço da web, o campo XOP_MODE de DFHWS-MTOM-OUT poderá ser alterado pelo CICS quando o contêiner for criado. Por exemplo, se você configurar o pipeline para processar o documento XOP e quaisquer anexos binários usando o modo direto e, em seguida, você ativar a validação para um serviço da web, o CICS substituirá o valor definido no arquivo de configuração de pipeline e configurará o processamento de XOP para executar no modo de compatibilidade quando ele criar o contêiner. O CICS executa a substituição devido a restrições no suporte para processar documentos XOP e anexos binários no pipeline.

Não é possível alterar o conteúdo deste contêiner.

Tabela 10. Estrutura do Contêiner DFHWS-MTOM-OUT

Nome do campo	Comprimento (bytes)	Conteúdo
MTOM_STATUS	4	Indica se o MTOM está ativado: 0 MTOM não está ativado. A mensagem de saída é enviada no formato SOAP. 1 MTOM está ativado. A mensagem de saída é enviada no formato MTOM.
MTOMNOXOP_STATUS	4	Indica se deve usar MTOM quando não há anexos binários: 0 Não enviar uma mensagem MTOM quando não há anexos binários. 1 Enviar uma mensagem MTOM quando não há anexos binários.
XOP_MODE	4	Indica qual processamento de XOP deve ocorrer: 0 Não ocorre nenhum processamento de XOP. 1 O processamento de XOP ocorre no modo de compatibilidade. 2 O processamento de XOP ocorre no modo direto.

Contêiner DFHWS-WSDL-CTX:

DFHWS-WSDL-CTX é um contêiner de DATATYPE(CHAR) que é usado em um aplicativo do provedor de serviços ou do solicitante de serviço implementado com o assistente de serviços da web do CICS. Ele contém informações de contexto de WSDL que podem ser usadas para propósitos de monitoramento.

O DFHWS-WSDL-CTX contém as informações de contexto a seguir para o documento WSDL:

- O nome e o namespace da operação para a qual o aplicativo está sendo chamado.
- Se conhecidos, o nome e o namespace para a porta WSDL 1.1 ou o terminal WSDL 2.0 que está sendo usado.

Estes valores são separados pelos caracteres de espaço. DFHWS-WSDL-CTX é preenchido pelo CICS somente no nível de tempo de execução 2.1 e mais recente.

Se você usou o assistente de serviços da web do CICS para implementar seu aplicativo, este contêiner será preenchido pelo CICS:

- Em um pipeline do provedor de serviços, este contêiner é preenchido pelo manipulador de aplicativo DFHPITP quando ele recebe a mensagem de entrada do manipulador de terminal.
- Em um pipeline do solicitante de serviço, este contêiner é preenchido quando o aplicativo usa o comando **INVOKE SERVICE**.

Se o aplicativo usar o programa DFHPIRT para iniciar o pipeline, o aplicativo preencherá o contêiner DFHWS-WSDL-CTX se necessário.

Contêiner DFHWS-XOP-IN:

DFHWS-XOP-IN é um contêiner de DATATYPE(BIT). Ele contém uma lista de referências aos anexos binários que foram descompactados de uma mensagem MIME de entrada e colocados em contêineres usando o processamento de XOP.

Cada registro de anexo no contêiner DFHWS-XOP-IN consiste nestes itens:

- O nome de 16 bytes do contêiner que contém os cabeçalhos MIME para o anexo binário
- O nome de 16 bytes do contêiner que contém o anexo binário
- O comprimento de 2 bytes do ID de conteúdo, no formato binário de meia-palavra sinalizado
- O ID de conteúdo, incluindo os delimitadores < e >, armazenados como uma sequência de caracteres ASCII

Não é possível alterar o conteúdo deste contêiner.

Contêiner DFHWS-XOP-OUT:

DFHWS-XOP-OUT é um contêiner de DATATYPE(BIT). Ele contém uma lista de referências aos contêineres que contêm anexos binários. Os anexos binários são empacotados em uma mensagem MIME de saída pelo programa manipulador MTOM.

Cada registro de anexo no contêiner DFHWS-XOP-OUT consiste nestes itens:

- O nome de 16 bytes do contêiner que contém os cabeçalhos MIME para o anexo binário
- O nome de 16 bytes do contêiner que contém o anexo binário
- O comprimento de 2 bytes do ID de conteúdo, no formato binário de meia-palavra sinalizado
- O ID de conteúdo, incluindo os delimitadores < e >, armazenados como uma sequência de caracteres ASCII

Não é possível alterar o conteúdo deste contêiner.

Os contêineres do programa de processamento de cabeçalho

Os manipuladores de mensagem SOAP 1.1 e SOAP 1.2 fornecidos pelo CICS se vinculam aos programas de processamento de cabeçalho usando o canal DFHHHC-V1. Os contêineres que são transmitidos no canal incluem vários que são específicos para a interface do programa de processamento de cabeçalho e conjuntos de *contêineres de contextos* e *contêineres do usuário* que são acessíveis para todos os programas de processamento de cabeçalho e programas do manipulador de mensagem no pipeline.

O contêiner DFHHEADER é específico para a interface do programa de processamento de cabeçalho. Outros contêineres estão disponíveis em outro lugar em seu pipeline, mas têm utilizações específicas em um programa de processamento de cabeçalho. Os contêineres nesta categoria são DFHWS-XMLNS, DFHWS-BODY e DFHXMLSS-PARSE.

Nota: Embora o serviço da web que usa Axis2 para processar mensagens SOAP possa usar a interface do programa de processamento de cabeçalho, é mais eficiente gravar seus próprios manipuladores Axis2 em Java para processar os

cabeçalhos SOAP. Para obter mais informações sobre como criar manipuladores Axis2, consulte Gravando seu Próprio Módulo Axis2

Contêiner DFHHEADER

Quando o programa de processamento de cabeçalho é chamado, DFHHEADER contém o bloco de cabeçalho único que faz com que o programa de processamento de cabeçalho fosse conduzido. Quando o programa do cabeçalho é especificado com `<mandatory>true</mandatory>` ou `<mandatory>1</mandatory>` no arquivo de configuração de pipeline, ele é chamado mesmo quando não há bloco de cabeçalho correspondente na mensagem SOAP. Neste caso, o contêiner DFHHEADER possui um comprimento igual a zero. Este é o caso quando um programa de processamento de cabeçalho é chamado para incluir um bloco de cabeçalho em uma mensagem SOAP que não possui blocos de cabeçalhos.

A mensagem SOAP que o CICS cria não possui cabeçalhos inicialmente. Se desejar incluir cabeçalhos em sua mensagem, você deverá assegurar que pelo menos um programa de processamento de cabeçalho seja chamado, especificando `<mandatory>true</mandatory>` ou `<mandatory>1</mandatory>`.

Quando o programa de cabeçalho retorna, o contêiner DFHHEADER deve conter zero, um ou mais blocos de cabeçalho que o CICS insere na mensagem SOAP no lugar do original:

- É possível retornar o bloco de cabeçalho original inalterado.
- É possível modificar o conteúdo do bloco de cabeçalho.
- É possível anexar um ou mais blocos de cabeçalho novos no bloco original.
- É possível substituir o bloco de cabeçalho original por um ou mais blocos diferentes.
- É possível excluir o bloco de cabeçalho completamente.

Contêiner DFHWS-XMLNS

Quando o programa de processamento do cabeçalho é chamado, DFHWS-XMLNS contém informações sobre namespaces XML que são declarados no envelope SOAP. O programa do cabeçalho pode usar estas informações para executar as tarefas a seguir:

- Resolver nomes qualificados que ele encontra no bloco de cabeçalho
- Construir nomes qualificados nos blocos de cabeçalho novos ou modificados.

As informações de namespace consistem em uma lista de declarações de namespace, que utilizam a notação XML padrão para declarar namespaces. As declarações de namespaces em DFHWS-XMLNS são separadas por espaços. Por exemplo:

```
xmlns:na='http://abc.example.org/schema' xmlns:nx='http://xyz.example.org/schema'
```

É possível incluir declarações de namespace adicionais no envelope SOAP anexando-as no conteúdo de DFHWS-XMLNS. No entanto, namespaces cujo escopo é um bloco de cabeçalho SOAP ou um corpo SOAP são melhor declarados no bloco de cabeçalho ou no corpo, respectivamente. É avisado para não excluir declarações de namespace do contêiner DFHWS-XMLNS em um programa de processamento de cabeçalho, porque os elementos XML que não são visíveis no programa podem depender delas.

Contêiner DFHWS-BODY

Este contêiner contém a seção do corpo do envelope SOAP. O programa de processamento de cabeçalho pode modificar o conteúdo.

Quando o programa de processamento de cabeçalho é chamado, o DFHWS-BODY contém o elemento SOAP <Body>.

Quando o programa de cabeçalho retorna, o contêiner DFHWS-BODY deve novamente conter um <Body> SOAP válido, que o CICS insere na mensagem SOAP no lugar do original:

- É possível retornar o corpo original inalterado.
- É possível modificar o conteúdo do corpo.

Você não deve excluir o corpo SOAP completamente, pois toda mensagem SOAP deve conter um elemento <Body>.

Contêiner DFHXMLSS-PARSE

Ao usar os elementos <cics_soap_1.1_handler> ou <cics_soap_1.2_handler> em sua configuração de pipeline, e o programa do cabeçalho é chamado, DFHXMLSS-PARSE contém os registros XML System Services (XMLSS) para esse cabeçalho. Este contêiner não é criado quando os elementos <cics_soap_1.1_handler_java> ou <cics_soap_1.2_handler_java> são usados.

Contêineres de Controle, Contexto e do Usuário

Assim como os contêineres descritos, a interface transmite os *contêineres de controle*, *contêineres de contextos* e *contêineres do usuário* no canal DFHHHC-V1.

Para obter mais informações sobre estes contêineres, consulte “Contêineres usados no pipeline” na página 148.

Contêineres de Segurança

Os contêineres de segurança são usados no canal DFHWSTC-V1 para enviar e receber tokens de identidade de um Serviço de Token de Segurança (STS) tal como o Tivoli Federated Identity Manager. Esta interface é chamada de *interface do cliente de Confiança* e pode ser usada em pipelines do solicitante e do provedor de serviço da web.

Contêiner DFHWS-IDTOKEN:

DFHWS-IDTOKEN é um contêiner de DATATYPE(CHAR). Ele contém o token que o Security Token Service (STS) valida ou usa para emitir um token de identidade para a mensagem.

O token deve estar no formato XML.

Use este contêiner somente com o canal DFHWSTC-V1 para a interface do cliente de Confiança.

Contêiner DFHWS-RESTOKEN:

DFHWS-RESTOKEN é um contêiner de DATATYPE(CHAR). Ele contém a resposta do Security Token Service (STS).

A resposta depende da ação que foi solicitada a partir do STS no contêiner DFHWS-STSACTION.

- Se a ação for emitir, este contêiner conterá o token que o STS trocou pelo que foi enviado no contêiner DFHWS-IDTOKEN.
- Se a ação for validar, este contêiner conterá um URI para indicar se o token de segurança que foi enviado no contêiner DFHWS-IDTOKEN é válido ou inválido. Os URIs que podem ser retornados são os seguintes:

URI	Descrição
http://schemas.xmlsoap.org/ws/2005/02/trust/status/valid	O token de segurança é válido.
http://schemas.xmlsoap.org/ws/2005/02/trust/status/invalid	O token de segurança é inválido.

Este contêiner é retornado no canal DFHWSTC-V1 ao usar a interface do cliente de Confiança.

Contêiner DFHWS-SERVICEURI:

DFHWS-SERVICEURI é um contêiner de DATATYPE(CHAR). Ele contém a URI que o Security Token Service (STS) usa como o escopo AppliesTo.

O escopo AppliesTo é usado para determinar o serviço da web com o qual o token de segurança está associado.

Use este contêiner somente com o canal DFHWSTC-V1 para a interface do cliente de Confiança.

Contêiner DFHWS-STSACTION:

DFHWS-STSACTION é um contêiner de DATATYPE(CHAR). Ele contém o URI da ação que o Security Token Service (STS) executa para validar ou emitir um token de segurança.

Os valores de URI que podem ser especificados neste contêiner são os seguintes:

URI	Descrição
http://schemas.xmlsoap.org/ws/2005/02/trust/Issue	O STS emite um token em troca por aquele que é enviado no contêiner DFHWS-IDTOKEN.
http://schemas.xmlsoap.org/ws/2005/02/trust/Validate	O STS valida o token que é enviado no contêiner DFHWS-IDTOKEN.

Use este contêiner apenas com o canal DFHWSTC-V1 para a interface do cliente de Confiança.

Contêiner DFHWS-STSFault:

DFHWS-STSFault é um contêiner de DATATYPE(CHAR). Ele contém o erro que foi retornado pelo Security Token Service (STS).

Se um erro ocorre, o STS emite uma falha de SOAP. O conteúdo da falha de SOAP é retornado neste contêiner.

Este contêiner é retornado no canal DFHWSTC-V1 ao usar a interface do cliente de Confiança.

Contêiner DFHWS-STSREASON:

DFHWS-STSREASON é um contêiner de DATATYPE(CHAR). Ele contém o conteúdo do elemento <wst:Reason>, se este elemento estiver presente na mensagem de resposta do Security Token Service (STS).

O elemento <wst:Reason> contém uma sequência opcional que fornece informações relacionadas ao status da solicitação de validação que foi enviada ao STS pelo CICS. Se o token de segurança não for válido, as informações fornecidas pelo STS neste elemento poderão ajudá-lo a determinar a razão pela qual o token não é válido.

Para obter mais informações, consulte a especificação *Web Services Trust Language* que é publicada em OASIS WS-Trust v1.4 Standard.

Contêiner DFHWS-STSURI:

DFHWS-STSURI é um contêiner de DATATYPE(CHAR). Ele contém o URI absoluto do Security Token Service (STS) que é usado para validar ou emitir um token de identidade para a mensagem SOAP.

O formato do URI é `http://www.example.com:8080/TrustServer/SecurityTokenService`. É possível usar HTTP ou HTTPS, dependendo de seus requisitos de segurança.

Use este contêiner somente com o canal DFHWSTC-V1 para a interface do cliente de Confiança.

Contêiner DFHWS-TOKENTYPE:

DFHWS-TOKENTYPE é um contêiner de DATATYPE(CHAR). Ele contém o URI do tipo de token solicitado que o Serviço de Token de Segurança (STS) emite como um token de identidade para a mensagem SOAP.

É possível especificar qualquer tipo de token válido, mas ele deve ser suportado pelo STS.

Use este contêiner somente com o canal DFHWSTC-V1 para a interface do cliente de Confiança.

Contêineres de suporte SAML

Os contêineres somente leitura que são usados pelo suporte SAML do CICS.

Nos tópicos a seguir, *nnn* significa que pode haver mais de um contêiner. Os contêineres são numerados de 001 a *nnn* (o número de contêineres deste tipo retornados). Mais de 999 contêineres de um tipo específico não são suportados e os dados na asserção SAML que está relacionada a eles são ignorados. Os contêineres que não são mapeados para um DSECT são de comprimento variável.

Contêiner DFHSAML-AnnnVmmm:

DFHSAML-AnnnVmmm é um contêiner de DATATYPE(CHAR). Ele contém o Valor de Atributo *mmm* para o atributo *nnn*, em que *nnn* e *mmm* são números de 3 dígitos.

O número de atributos é SAMLC-ATTRNUM no contêiner DFHSAML-COUNTS.

O número de valores para esse atributo está em DFHSAML-ATTRAnnn.

Contêiner DFHSAML-ASSQNAME:

DFHSAML-ASSQNAME é um contêiner de DATATYPE(CHAR). Ele contém o namespace de Asserção SAML.

Os valores possíveis são

SAML 1.1

urn:oasis:names:tc:SAML:1.0:assertion

SAML 2.0

urn:oasis:names:tc:SAML:2.0:assertion

Esta asserção deve ser um URI. Se a asserção for mais complexa, ela será extraída em 3 partes.

Contêiner DFHSAML-ATTRAnnn:

DFHSAML-ATTRAnnn é um contêiner de DATATYPE(BIT). Ele contém um campo BIN(31) com o número de valores para o atributo *nnn*. O número máximo de valores é 999.

O número de atributos é SAMLC-ATTRNUM no contêiner DFHSAML-COUNTS.

Contêiner DFHSAML-ATTRFnnn:

DFHSAML-ATTRFnnn é um contêiner de DATATYPE(CHAR). Ele contém o formato de nome do Atributo para o atributo *nnn*, em que *nnn* é um número de 3 dígitos.

O número de atributos é SAMLC-ATTRNUM no contêiner DFHSAML-COUNTS.

Contêiner DFHSAML-ATTRNnnn:

DFHSAML-ATTRNnnn é um contêiner de DATATYPE(CHAR). Ele contém o Nome do Atributo para o atributo *nnn*, em que *nnn* é um número de 3 dígitos.

O número de atributos é SAMLC-ATTRNUM no contêiner DFHSAML-COUNTS.

Contêiner DFHSAML-ATTRSnnn:

DFHSAML-ATTRSnnn é um contêiner de DATATYPE(CHAR). Ele contém o Espaço de Nome do Atributo para o atributo *nnn*, em que *nnn* é um número de 3 dígitos.

O número de atributos é SAMLC-ATTRNUM no contêiner DFHSAML-COUNTS.

Contêiner DFHSAML-ATTRYnnn:

DFHSAML-ATTRYnnn é um contêiner de DATATYPE(CHAR). Ele contém o nome fácil do Atributo para o atributo *nnn*, em que *nnn* é um número de 3 dígitos.

O número de atributos é SAMLC-ATTRNUM no contêiner DFHSAML-COUNTS.

Contêiner DFHSAML-AUDNRnnn:

DFHSAML-AUDNRnnn é um contêiner de DATATYPE(CHAR). Ele contém o nome de AudienceRestriction.

O número de contêineres retornados é AUDNRNUM.

Contêiner DFHSAML-AUTHMETH:

DFHSAML-AUTHMETH é um contêiner de DATATYPE(CHAR). Ele contém o método que é usado para autenticar o portador do token.

Os métodos incluem password, Kerberos e ltpa.

Contêiner DFHSAML-CERTIDN:

DFHSAML-CERTIDN é um contêiner de DATATYPE(CHAR). Ele contém o nome distinto do emissor do Certificado X.509 do assinante SAML.

Contêiner DFHSAML-CERTSDN:

DFHSAML-CERTSDN é um contêiner de DATATYPE(CHAR). Ele contém o nome distinto do assunto do certificado X.509 do assinante SAML.

Contêiner DFHSAML-CERTSNUM:

DFHSAML-CERTSNUM é um contêiner de DATATYPE(CHAR). Ele é um campo de oito caracteres que contém o número de série do Certificado X.509 do assinante SAML.

Contêiner DFHSAML-CONFMETH:

DFHSAML-CONFMETH é um contêiner de DATATYPE(CHAR). Ele contém o método SubjectConfirmation que é usado neste token SAML.

Os métodos válidos são holder-of-key, bearer ou sender-vouches. A sequência retornada é baseada no OASIS SAML Token Profile 1.1 e 2.0.

Nota: Tokens SAML que possuem mais de um método de confirmação não são suportados. Se houver mais de um método de confirmação, os resultados serão imprevisíveis.

Contêiner DFHSAML-COUNTS:

DFHSAML-COUNTS é um contêiner de DATATYPE(BIT). Ele contém o número de contêineres de comprimento variável enviados.

Contêiner DFHSAML-FLAGS:

DFHSAML-FLAGS é um contêiner de DATATYPE(Char). Ele contém uma coleção de bytes sinalizadores.

Contêiner DFHSAML-ISSUER:

DFHSAML-ISSUER é um contêiner de DATATYPE(Char). Ele contém o nome do emissor.

Contêiner DFHSAML-NAMID:

DFHSAML-NAMID é um contêiner de DATATYPE(Char). Ele contém o valor da propriedade de formato de nome.

Contêiner DFHSAML-NAMIDF:

DFHSAML-NAMIDF é um contêiner de DATATYPE(Char). Ele contém uma referência de URI que representa a classificação de informações do identificador baseado em sequência.

Contêiner DFHSAML-NAMIDQ:

DFHSAML-NAMIDQ é um contêiner de DATATYPE(Char). Ele contém o domínio de segurança ou administrativo que qualifica o nome.

Contêiner DFHSAML-NAMIDSP:

DFHSAML-NAMIDSP é um contêiner de DATATYPE(Char). Ele contém o identificador de nome que é estabelecido por um provedor de serviços ou afiliação de provedores da entidade.

Contêiner DFHSAML-NAMIDSPQ:

DFHSAML-NAMIDSPQ é um contêiner de DATATYPE(Char). Ele contém o nome de um provedor de serviços ou afiliação de provedores.

Contêiner DFHSAML-OUTTOKEN:

DFHSAML-OUTTOKEN é um contêiner de DATATYPE(Char). Ele contém um token SAML.

Se este é um contêiner de entrada, ele contém o token validado anteriormente, que está sendo roteado para outro provedor de serviços ou estendido e, em seguida, roteado.

Se este é um contêiner de saída, ele contém uma saída token SAML por processamento DFHSAML. Se o processamento for validação ou extração, esse token será validado, extraído ou modificado e renunciado.

Contêiner DFHSAML-PROXYnnn:

DFHSAML-PROXYnnn é um contêiner de DATATYPE(Char). Ele contém o nome do Público de ProxyRestriction.

Contêiner DFHSAML-RESPONSE:

DFHSAML-RESPONSE é um contêiner de DATATYPE(BIT). Ele contém um código de resposta que é usado internamente.

Contêiner DFHSAML-SAMLID:

DFHSAML-SAMLID é um contêiner de DATATYPE(CHAR). Ele contém uma sequência que representa o ID para SAML 2.0 ou AssertionID para SAML 1.1.

Contêiner DFHSAML-SUBJADDR:

DFHSAML-SUBJADDR é um contêiner de DATATYPE(CHAR). Ele contém o endereço IP em SubjectLocality.

Restrição: Este contêiner não é retornado para SAML 2.0.

Contêiner DFHSAML-SUBJDNS:

DFHSAML-SUBJDNS é um contêiner de DATATYPE(CHAR). Ele contém o DNSAddress em SubjectLocality.

Contêiner DFHSAML-TIMES:

DFHSAML-TIMES é um contêiner de DATATYPE(CHAR). Ele contém uma coleção de valores de hora.

Contêineres Gerados pelo CICS

O CICS gera contêineres para armazenar dados tais como matrizes de variável e sequências longas. Estes contêineres são criados durante o processamento de pipeline e são usados como entrada ou saída do programa de aplicativo. Estes contêineres são prefixados com DFH.

A convenção de nomenclatura para esses contêineres é usar o módulo CICS que os criou, combinado com um sufixo numérico para tornar o nome do contêiner exclusivo na solicitação. Estes nomes do contêiner ocorrem durante o processamento de pipeline:

DFHPIAXIS-*nnnnnnnn*

Os contêineres que são usados para armazenar sequências e matrizes de variável que são transmitidas ao aplicativo nos pipelines Axis2. Este contêiner também pode incluir dados binários.

DFHPICC-*nnnnnnnnnn*

Os contêineres que são usados para armazenar sequências e matrizes de variável que são transmitidas ao aplicativo. Este contêiner também pode incluir dados binários.

DFHPIII-*nnnnnnnnnn*

Contêineres do anexo de saída criados quando o pipeline é ativado com o manipulador de mensagem MTOM e está em execução no modo direto. Estes contêineres são criados quando dados binários são fornecidos em um campo em vez de em um contêiner pelo programa de aplicativo.

DFHPIMM-*nnnnnnnnnn*

Os contêineres do anexo de entrada criados durante o processamento de mensagens MIME. Estes contêineres são gerados pelo CICS quando o manipulador de mensagem MTOM é ativado no pipeline. Quando o

processamento de modo direto é ativado, estes contêineres podem ser transmitidos ao aplicativo diretamente.

DFHPIIXO-xxxxxxxxxx

Os contêineres do anexo de saída criados quando o pipeline é ativado com o manipulador de mensagem MTOM e está em execução no modo de compatibilidade.

Os nomes do contêiner numerados iniciam em 1 para cada solicitação de serviço da web; por exemplo, DFHPICC-00000001. No entanto, se um programa de aplicativo usa o comando **INVOKE SERVICE** para iniciar mais de uma solicitação de serviço da web no mesmo canal, os contêineres que foram retornados para o aplicativo para uma resposta ainda poderão existir quando uma solicitação adicional é feita. Nesta situação, o CICS verifica se o contêiner já existe e incrementa o número do contêiner gerado para evitar um conflito de nomenclatura.

Contêineres do Usuário

Estes contêineres contêm informações que um manipulador de mensagem precisa transmitir para um outro. O uso de contêineres de usuário é inteiramente uma questão para os manipuladores de mensagens. É possível escolher seus próprios nomes para estes contêineres, mas você não deve usar nomes que iniciam com DFH.

Processamento de Tempo de Execução para Serviços da Web

Para enviar uma solicitação para um provedor de serviço da web ou para receber uma solicitação de um solicitante de serviço da web, seu aplicativo (ou programa wrapper) deve interagir corretamente com o suporte de serviços da web no CICS. Também é possível controlar o processamento que ocorre no pipeline para determinar como as solicitações de entrada e saída são manipuladas.

Como o CICS Chama um Programa do Provedor de Serviços Implementado com o Assistente de Serviços da Web

Quando um aplicativo do provedor de serviços que foi implementado usando o assistente de serviços da web do CICS é chamado, o CICS se vincula a ele com uma COMMAREA ou um canal.

Você especifica qual tipo de interface é usada ao executar o procedimento da JCL DFHWS2LS ou DFHLS2WS com o parâmetro **PGMINT**. Se especificar um canal, será possível nomear o contêiner no parâmetro **CONTID**.

- Se o programa for chamado com uma interface de COMMAREA, a COMMAREA conterá a estrutura de dados de nível superior que o CICS criou a partir da solicitação de SOAP.
- Se o programa for chamado com uma interface do canal, a estrutura de dados de nível superior será transmitida ao seu programa no contêiner que foi especificado no parâmetro **CONTID** de DFHWS2LS ou DFHLS2WS. Se você não especificou o parâmetro **CONTID**, os dados serão transmitidos no contêiner DFHWS-DATA. A interface do canal suporta matrizes com números variados de elementos, que são representados como séries de estruturas de dados conectadas em uma série de contêineres. Estes contêineres também estarão presentes.

Ao codificar comandos de API para trabalhar com os contêineres, não é necessário especificar a opção **CHANNEL**, porque todos os contêineres são associados ao canal atual (o canal que foi transmitido ao programa). Se precisar saber o nome do canal, use o comando **EXEC CICS ASSIGN CHANNEL**.

Quando seu programa tiver processado a solicitação, ele deverá usar o mesmo mecanismo para retornar a resposta: se a solicitação foi recebida em uma COMMAREA, a resposta deverá ser retornada na COMMAREA; se a solicitação foi recebida em um contêiner, a resposta deverá ser retornada no mesmo contêiner.

Se um erro for encontrado quando o programa de aplicativo estiver emitindo uma mensagem de resposta, o CICS recuperará todas as mudanças, a menos que o aplicativo tenha executado um ponto de sincronização.

Se o serviço da web fornecido por seu programa não foi projetado para retornar uma resposta, o CICS ignorará qualquer coisa na COMMAREA ou no contêiner quando o programa retornar.

Chamando um Serviço da Web a Partir de um Aplicativo Implementado com o Assistente de Serviços da Web

Um aplicativo do solicitante de serviço que é implementado com o assistente de serviços da web usa o comando **EXEC CICS INVOKE SERVICE** para chamar um serviço da web. A solicitação e a resposta são mapeadas para uma estrutura de dados no contêiner DFHWS-DATA. Esse método de chamar um serviço não é suportado por JSON.

Procedimento

1. Crie um canal e o preencha com contêineres. No mínimo, o contêiner DFHWS-DATA deve estar presente. DFHWS-DATA contém a estrutura de dados de nível superior que o CICS converterá em uma solicitação SOAP. Se a solicitação SOAP contiver quaisquer matrizes que possuem números variados de elementos, eles serão representados como uma série de estruturas de dados conectadas em uma série de contêineres. Estes contêineres também devem estar presentes no canal.
2. Chame o serviço da web de destino. Utilize o comando a seguir:

```
EXEC CICS INVOKE SERVICE(webservice)  
                  CHANNEL(userchannel)  
                  OPERATION(operation)
```

em que:

- *webservice* é o nome do recurso WEBSERVICE que define o serviço da web a ser chamado. O recurso WEBSERVICE especifica o local da descrição de serviços da web e do arquivo de ligação de serviço da web que o CICS usa quando ele se comunica com o serviço da web.
- *userchannel* é o canal que contém o contêiner DFHWS-DATA e quaisquer outros contêineres associados à estrutura de dados do aplicativo.
- *operation* é o nome da operação que deve ser chamada no serviço da web de destino.

Para obter mais informações, consulte “Otimização local para serviços da web” na página 183.

3. Se o comando foi bem-sucedido, recupere os contêineres de resposta a partir do canal. No mínimo, o contêiner DFHWS-DATA estará presente. Ele contém a estrutura de dados de nível superior que o CICS criou a partir da resposta SOAP. Se a resposta contiver qualquer matriz que possua números variados de elementos, eles serão representados como uma série de estruturas de dados conectadas em uma série de contêineres. Estes contêineres estarão presentes no canal.

4. Se o solicitante de serviço receber uma mensagem de falha de SOAP do serviço da web chamado, você deverá decidir se o programa de aplicativo deve recuperar qualquer mudança. Se ocorrer uma falha de SOAP, um erro de INVREQ com um valor de RESP2 igual a 6 será retornado ao programa de aplicativo. No entanto, se a otimização estiver em efeito, a mesma transação será usada no solicitante e no provedor. Se ocorrer um erro em um provedor de serviço da web otimizado localmente, todo o trabalho feito pela transação será recuperado no provedor e no solicitante. Um erro de INVREQ é retornado ao solicitante com um valor de RESP2 igual a 16.

Otimização local para serviços da web

É possível usar o nome do aplicativo do provedor no arquivo de ligação de serviço da web associado ao recurso WEBSERVICE para ativar a otimização local da solicitação de serviço da web.

Usando o comando INVOKE SERVICE, é possível especificar o URIMAP(urimap) ou URI(uri) em que o uri é o URI do serviço da web a ser chamado. Se um URIMAP for especificado, o CICS usará o URIMAP do modo cliente indicado para resolver o URI. Se estas opções forem omitidas, o CICS usará o URI especificado na descrição do serviço da web (WSDL) a partir da qual o WEBSERVICE foi gerado.

Se o WEBSERVICE indicado estiver implementado em um PIPELINE no modo do solicitante, o CICS chamará o serviço da web remoto. Este é o cenário mais comum.

Se o WEBSERVICE indicado for implementado em um PIPELINE no modo de provedor, o CICS chamará o serviço localmente. Se você usar esta otimização, o comando EXEC CICS INVOKE SERVICE será otimizado para um comando EXEC CICS LINK. Isso resulta em benefícios de desempenho significativos, mas apresenta as seguintes limitações:

- O PIPELINE não é acionado e, portanto, nenhum programa manipulador é usado.
- Os contêineres de controle do PIPELINE não estão presentes no Canal. Alguns contêineres estão presentes, incluindo os contêineres DFHWS-DATA, DFHWS-OPERATION e DFHWS-URI. Quaisquer contêineres que normalmente contêm XML não estão presentes; isso inclui os contêineres DFH-REQUEST, DFHWS-BODY e DFHWS-XMLNS.
- Ambos os aplicativos, do provedor e do solicitante, devem compartilhar os mesmos copybooks e serem implementados na mesma linguagem de programação.
- Ambos os aplicativos, do provedor e do solicitante, compartilham uma única unidade de trabalho.
- Se não espera-se que o serviço da web retorne uma resposta, o comando EXEC CICS INVOKE SERVICE não retorna o controle para o aplicativo até depois do encerramento do PROGRAM de destino.

Se desejar usar serviços da web otimizados localmente, mas requerer que dados sejam processados por meio de um PIPELINE, use o formato de URI cics descrito aqui: “Opções para Controlar o Processamento do Pipeline do Solicitante” na página 188. Esse mecanismo é menos eficiente do que utilizar a abordagem totalmente otimizada, mas evita o custo de processamento de sair para a rede.

Limitações de Tempo de Execução para Código Gerado pelo Assistente de Serviços da Web

No tempo de execução, o CICS é capaz de transformar quase toda mensagem SOAP válida que está em conformidade com a descrição de serviços da web (WSDL) nas estruturas de dados equivalentes. No entanto, há algumas limitações das quais você deve estar ciente ao desenvolver um aplicativo do solicitante de serviço ou do provedor de serviços usando as tarefas em lote do assistente de serviços da web.

Páginas de Códigos

O CICS pode suportar mensagens SOAP enviadas a ele em qualquer página de códigos se houver um cabeçalho HTTP ou WebSphere MQ apropriado que identifica a página de códigos. O CICS converte a mensagem SOAP em UTF-8 para processá-la no pipeline, antes de transformá-la na página de códigos requerida pelo programa de aplicativo. Para minimizar o impacto do desempenho, é recomendado que você use a página de códigos UTF-8 ao enviar mensagens SOAP ao CICS. O CICS sempre envia mensagens SOAP em UTF-8.

O CICS pode transformar mensagens SOAP somente se os dados do aplicativo forem codificados em EBCDIC ou UTF-16. Os aplicativos que esperam que os dados sejam codificados em páginas de códigos como UTF-8, ASCII e ISO8859-1 não são suportados. Se desejar usar caracteres DBCS dentro de suas estruturas de dados e mensagens SOAP, você deverá especificar uma página de códigos que suporta DBCS. A página de códigos EBCDIC selecionada também deve ser suportada pelos serviços de conversão Java e z/OS. Os serviços de conversão de z/OS também devem ser configurados para suportar a conversão da página de códigos da mensagem SOAP em UTF-8. Consulte Suporte para UTF-16 em dados do aplicativo para obter mais informações sobre o suporte UTF-16.

Para configurar uma página de códigos apropriada, é possível usar o parâmetro de inicialização do sistema **LOCALCCSID** ou o parâmetro **CCSID** opcional nas tarefas do assistente de serviços da web. Se você usar o parâmetro **CCSID**, o valor especificado substituirá a página de códigos **LOCALCCSID** para esse serviço da web específico. Se não especificar o parâmetro **CCSID**, a página de códigos **LOCALCCSID** será usada para converter os dados e o arquivo de ligação de serviço da web será codificado em US EBCDIC (Cp037).

Containers

No modo do provedor de serviços, se você especificar que o parâmetro **PGMINT** possui um valor igual a **CHANNEL**, o contêiner que contém seus dados do aplicativo deverão ser gravados e lidos no modo binário. Este contêiner é DFHWS-DATA por padrão. O comando **PUT CONTAINER** deve-se ter a opção **DATATYPE** configurada como **BIT** ou você deve omitir a opção **FROMCCSID** para que **BIT** permaneça o padrão. Por exemplo, o código a seguir indica explicitamente que os dados no contêiner **CUSTOMER-RECORD** no canal atual devem ser gravados no modo binário.

```
EXEC CICS PUT CONTAINER (CUSTOMER-RECORD)
          FROM (CREC)
          DATATYPE(BIT)
```

Embora os próprios contêineres estejam todos no modo **BIT**, quaisquer campos de texto na estrutura de linguagem que mapeiam estes dados deverão usar uma página de códigos EBCDIC - a mesma página de códigos que você especificou no parâmetro **LOCALCCSID** ou **CCSID**. Se estiver usando **DFHWS2LS** para gerar o arquivo de ligação de serviço da web, poderá haver muitos contêineres no canal

que contêm partes da estrutura de dados completa. Se este for o caso, os campos de texto em cada um destes contêineres precisarão ser lidos e gravados usando a mesma página de códigos.

Se o programa de aplicativo estiver preenchendo contêineres que serão convertidos em mensagens SOAP, o aplicativo será responsável por assegurar que os contêineres tenham a quantidade correta de conteúdo. Se um contêiner contiver menos dados do que o esperado, o CICS emitirá um erro de conversão.

Se um programa de aplicativo usar o comando **INVOKE SERVICE**, quaisquer contêineres que ele transmitir ao CICS poderão potencialmente ser reutilizados e os dados dentro deles substituídos. Se desejar manter os dados nestes contêineres, crie um novo canal e copie os contêineres nele antes de executar o programa. Se você tiver um serviço da web de modo de provedor que também é um serviço da web do modo do solicitante, é recomendado que use um canal diferente ao usar o comando **INVOKE SERVICE**, em vez de usar o canal padrão ao qual ele foi anexado originalmente. Se seu programa de aplicativo estiver usando o comando **INVOKE SERVICE** várias vezes, é recomendado que você use canais diferentes em cada chamada para CICS ou assegure que todos os dados importantes da primeira solicitação sejam salvos antes de fazer a segunda solicitação.

Conformidade com a Descrição de Serviços da Web

Uma descrição de serviços da web pode descrever alguns dos conteúdos possíveis de uma mensagem SOAP como opcionais. Se este for o caso, DFHWS2LS alocará campos dentro da estrutura de linguagem gerada para indicar se o conteúdo está presente ou não. No tempo de execução, o CICS preenche estes campos de acordo. Se um campo, por exemplo um sinalizador de existência ou um campo de ocorrência, indicar que as informações não estão presentes, o programa de aplicativo não deverá tentar processar os campos associados a esse conteúdo opcional.

Se uma mensagem SOAP não tiver algum de seu conteúdo quando o CICS transformá-la, os campos equivalentes dentro das estruturas de dados não serão inicializados quando transmitidos ao programa de aplicativo.

Uma descrição de serviços da web também pode especificar as regras de processamento de espaço em branco a serem usadas ao ler uma mensagem SOAP e o CICS implementa estas regras no tempo de execução.

- Se o valor do aspecto `xsd:whiteSpace` for `replace`, os caracteres de espaço em branco tais como “tab” e “retorno de linha” serão substituídos por espaços.
- Se o valor do aspecto `xsd:whiteSpace` for `collapse`, qualquer caractere de espaço em branco final será removido ao gerar mensagens SOAP. No tempo de execução, mensagens SOAP de entrada são analisadas de acordo com a especificação de Esquema XML e todos os espaços em branco iniciais, finais e integrados são removidos.

mensagens SOAP

O CICS não suporta derivação de conteúdo da mensagem SOAP. Por exemplo, uma mensagem SOAP poderia usar o atributo `xsi:type` para especificar que um elemento possui um tipo específico, juntamente com um atributo `xsi:schemaLocation` para especificar o local do esquema que descreve o elemento. O CICS não suporta a capacidade de recuperar dinamicamente o esquema e transformar o valor do elemento com base no conteúdo do esquema. O CICS

suporta o atributo `xsi:nil` quando o nível de mapeamento configurado no assistente de serviço da web é 1.1 ou superior, mas este é o único atributo de instância do esquema XML que é suportado.

DFHWS2LS pode precisar fazer suposições sobre o comprimento ou tamanho máximo de alguns valores na mensagem SOAP. Por exemplo, se o esquema XML não especificar um comprimento máximo para um `xsd:string`, DFHWS2LS assumirá que o comprimento máximo é de 255 caracteres e gerará uma estrutura de linguagem de acordo. É possível alterar este valor usando o parâmetro **DEFAULT-CHAR-MAXLENGTH** em DFHWS2LS. No tempo de execução, se o CICS encontrar uma mensagem SOAP com um valor que é maior do que o espaço que foi alocado na estrutura de linguagem, o CICS emitirá um erro de conversão.

Se o CICS for o provedor de serviços, uma mensagem de falha de SOAP será retornada ao solicitante. Se o CICS for o solicitante de serviço, um código RESP2 apropriado será retornado do comando **INVOKE SERVICE**.

Alguns caracteres possuem mapeamentos especiais no XML, tal como os caracteres `<` e `>`. Se qualquer um destes caracteres especiais aparecerem dentro de uma matriz de caracteres que é processada pelo CICS no tempo de execução, ela será substituída pela entidade equivalente. As entidades XML que são suportadas são:

Caractere	Entidade XML
&	&
<	<
>	>
"	"
'	'

O CICS também suporta as formas canônicas das referências de caractere numérico usadas para códigos de espaço em branco:

Caractere	Entidade XML
Tabular		
Retorno de carro	

Alimentação de linha	

Observe que este suporte não se estende a nenhum programa manipulador de pipeline que é chamado.

O caractere nulo (x '00') é inválido em qualquer documento XML. Se um campo de tipo de caractere que é fornecido pelo programa de aplicativo contiver o caractere nulo, o CICS trancará os dados nesse ponto. Isto permite tratar matrizes de caracteres como sequências terminadas em nulo. Os campos de tipo de caractere gerados por DFHWS2LS a partir de tipos de dados de esquema XML `base64Binary` ou `hexBinary` representam dados binários e poderão conter caracteres nulos sem truncamento.

Atenção: O CICS gera dados XML e JSON a partir de dados de aplicativos estruturados. Se esses dados do aplicativo contêm padrões de bits que parecem XML, JSON, HTML, imagens JPEG ou qualquer outro tipo de conteúdo significativo pré-formatado, o CICS não conhece o significado semântico e processa esses dados como texto ou dados binários comuns. O CICS não tenta reconhecer padrões nos dados ou processar dados codificados de forma diferente. Por exemplo, se os dados contêm XML pré-formatado (tal como texto codificado em CDATA), esses dados são processados da mesma maneira que qualquer outro dado. Considere os dados do aplicativo a seguir: "An example: <here>". Este exemplo de dados fornecidos pelo aplicativo contém o que parece ser uma tag XML, mas ele será processado como texto bruto, resultando na representação XML a seguir: "An example: < here > ". Se um aplicativo precisar gerar XML ele mesmo, considere usar construções xsd:any em seus esquemas XML ou usar XML-ONLY=TRUE nos Assistentes.

Mensagens de Falha de SOAP

Se o CICS for o provedor de serviços, e você desejar que o programa de aplicativo emita uma mensagem de falha de SOAP, use o comando **SOAPFAULT CREATE**. Para usar este comando de API, você deve especificar que o parâmetro **PGMINT** do assistente de serviços da web possui um valor igual a **CHANNEL**. Se você não especificar este valor, e o programa de aplicativo chamar o comando **SOAPFAULT CREATE**, o CICS não tentará gerar uma mensagem de resposta SOAP.

Customizando o Processamento de Pipeline

Além de fornecer seus próprios manipuladores de mensagens, você também pode utilizar um conjunto de pontos de saída do usuário global (GLUEs) para customizar o processamento que ocorre para serviços da web de entrada e de saída no pipeline.

Antes de Iniciar

Você deve compreender as melhores práticas para gravar programas de saída de usuário global antes de customizar o pipeline. Se estiver customizando um pipeline do provedor de serviços, você deverá estar utilizando o manipulador de aplicativo DFHPITP ou Axis2 fornecido em seu pipeline.

Sobre Esta Tarefa

É possível utilizar as saídas de domínio de pipeline para acessar contêineres em um pipeline do provedor de serviços da web, um pipeline do solicitante de serviços da web ou um pipeline do solicitante de serviços da web que contém um manipulador de segurança. As saídas de usuário global de pipeline são descritas em detalhes em Saídas de Domínio do Pipeline.

Procedimento

1. Selecione quais pontos de saída do usuário global utilizar:
 - Utilize os GLUEs XWSPRRWI, XWSPRROI, XWSPRROO ou XWSPRRWO para acessar os contêineres em um pipeline do provedor de serviços da web.
 - Utilize os GLUEs XWSRQRWO, XWSRQROO, XWSROROI or XWSRQRWI para acessar contêineres em um pipeline do solicitante de serviços da web.
 - Utilize os GLUEs XWSSRRWO, XWSSRROO, XWSSRROI ou XWSSRRWI para acessar contêineres em um pipeline do solicitante de serviços da web protegido.

2. Use o programa de saída de amostra DFH\$PIEX para gravar seu próprio programa de saída do usuário global. DFH\$PIEX está na biblioteca SDFHSAMP. É recomendado tornar o programa thread-safe.
3. Ative o programa de saída de usuário global.
4. Teste seu programa de saída de usuário global para assegurar que ele funciona corretamente.

Opções para Controlar o Processamento do Pipeline do Solicitante

Em pipelines do solicitante de serviço, os manipuladores de mensagem podem determinar onde a solicitação de serviço da web é enviada alterando o URI. O CICS fornece suporte para diferentes formatos de URI para que você tenha muito mais flexibilidade na maneira como o pipeline processa solicitações de serviço da web.

Quando o pipeline do solicitante de serviço atingir o final do processamento, você terá as seguintes opções:

Vincular a um programa

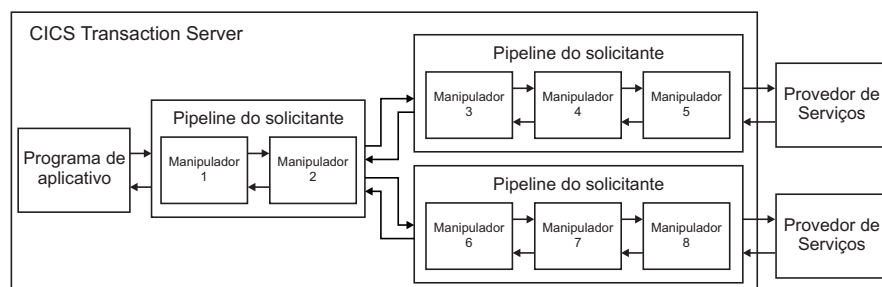
Se você alterar o URI para o formato `cics://PROGRAM/program`, em que *program* é o nome do programa de aplicativo de destino, o CICS passará o canal atual e seus contêineres ou o COMMAREA para o programa usando um comando **EXEC CICS LINK**.

Esse processamento é semelhante à otimização local que acontece quando o solicitante de serviço e os aplicativos do provedor de serviço estiverem na mesma região do CICS. Porém, usar esse formato de URI tem a vantagem de executar o pedido primeiro através do pipeline e de quaisquer manipuladores de mensagem customizados. O programa de aplicativo de destino deve poder manipular o conteúdo dos contêineres ou do COMMAREA.

Iniciando outro pipeline no modo solicitante

Se você alterar o URI para o formato `cics://PIPELINE/pipeline?targetServiceUri=targetServiceUri`, em que *pipeline* é o nome de um recurso de PIPELINE e *targetServiceUri* é a URI na qual deseja colocar o contêiner DFHWS-URI, o CICS passará o canal atual e seus contêineres para o pipeline do solicitante especificado. Use este URI quando desejar vincular dois ou mais pipelines do solicitante antes de enviar o pedido para o provedor de serviço. O número de pipelines do solicitante que você pode encadear não é limitado.

No exemplo a seguir, um pipeline do solicitante genérico suporta um aplicativo. Os manipuladores de mensagens 1 ou 2 podem alterar o URI para cada solicitação dependendo dos dados do aplicativo nos contêineres, enviando a solicitação para um dos dois pipelines do solicitante que contêm manipuladores de mensagens diferentes.



Embora o exemplo mostre somente um aplicativo solicitante de serviço, muitos aplicativos poderão usar o mesmo pipeline do solicitante genérico e ter suas solicitações enviadas para pipelines do solicitante diferentes antes da solicitação ser finalmente enviada ao provedor de serviços da web apropriado.

Enviando o pedido diretamente para o pipeline do modo do provedor

Se você alterar a URI para o formato `cics://SERVICE/service?targetServiceUri=targetServiceUri`, em que *service* é o nome do serviço de destino e *targetServiceUri* é o caminho para o serviço, o CICS resolverá o pedido ao corresponder o caminho para um URIMAP e passará o pedido para o pipeline do provedor correto. Use esta opção quando desejar obter vantagem do processamento do pedido através dos pipelines do solicitante e do provedor sem usar a rede.

Este URI também pode ser útil onde os aplicativos solicitante e provedor forem gravados em linguagens diferentes ou utilizarem níveis de mapeamento diferentes e esperarem dados binários diferentes.

Controlando o Processamento do Pipeline do Solicitante Usando um URI

Em pipelines do solicitante de serviço, um manipulador de mensagem pode determinar onde enviar a solicitação de serviço da web alterando o URI. Alterando o formato do URI, é possível escolher executar determinadas otimizações, tal como iniciar um outro pipeline do solicitante ou iniciar um pipeline do provedor de serviços sem enviar a solicitação pela rede.

Antes de Iniciar

Decida quais opções deseja implementar em seu pipeline do solicitante. Consulte “Opções para Controlar o Processamento do Pipeline do Solicitante” na página 188 para obter detalhes.

Sobre Esta Tarefa

O aplicativo do solicitante de serviço da web pode preencher o contêiner DFHWS-URI usando o comando **EXEC CICS INVOKE SERVICE** ou, se nenhum valor for fornecido pelo aplicativo, o CICS preencherá o contêiner usando o valor no arquivo de ligação do serviço da web. Para modificar o URI, você deve gravar um manipulador de mensagem que altera o conteúdo deste contêiner.

Procedimento

1. Grave um manipulador de mensagem para modificar o contêiner DFHWS-URI de acordo com um dos formatos de URI a seguir:
 - Para vincular-se a um programa de aplicativo, use o URI `cics://PROGRAM/program`, em que *program* é o programa de aplicativo de destino. Não ocorre nenhuma transformação de dados, portanto, você deve assegurar que o programa de aplicativo possa processar o conteúdo dos contêineres no canal atual. O programa de aplicativo pode transmitir o atual e os contêineres atuais ou uma COMMAREA.
 - Para iniciar um pipeline do provedor sem passar pela rede, use o URI `cics://SERVICE/service?targetServiceUri=targetServiceUri`, em que *service* é o nome do serviço e *targetServiceUri* é o caminho do serviço. O manipulador de transporte usa o caminho do serviço para localizar o recurso URIMAP que resolve a solicitação e a transmite ao pipeline do provedor correto. O CICS não usa o nome do serviço em seu processamento.

Ocorre um erro se nenhum recurso URIMAP é instalado para o serviço. A definição de recurso URIMAP também deve especificar USAGE(PIPELINE). O manipulador de transporte coloca o valor do parâmetro **targetServiceUri** no contêiner DFHWS-URI e inicia o pipeline do provedor.

- Para iniciar um outro pipeline do solicitante, use o URI `cics://PIPELINE/pipeline?targetServiceUri=targetServiceUri`, em que *pipeline* é o nome do recurso PIPELINE que você deseja iniciar e *targetServiceUri* é o valor que você deseja transmitir ao próximo pipeline no contêiner DFHWS-URI.

Cada tipo de URI possui parâmetros adicionais que podem ser incluídos como uma sequência de consultas. Para obter mais informações sobre o formato destes URIs e as regras para codificá-los, consulte “Contêiner DFHWS-URI” na página 164.

2. Use um editor XML para incluir o manipulador de mensagem no arquivo de configuração de pipeline:

```
<service>
  <service_handler_list>
    <handler>
      <program>MYPROG</program>
    </handler>
  </service_handler_list>
</service>
```

3. Desative, descarte e reinstale o recurso PIPELINE para o pipeline do solicitante para incluir seu novo programa do manipulador de mensagem no pipeline.
4. Instale o programa do manipulador de mensagem na região do CICS.

Resultados

A próxima solicitação de serviço a ser executada por meio do pipeline do solicitante é processada por seu novo manipulador de mensagem.

O que Fazer Depois

Teste as mudanças em seu pipeline do solicitante para certificar-se que as solicitações de serviço estejam indo para o local correto e que seu programa do manipulador de mensagem esteja se comportando conforme projetado.

Suporte para Transações de Serviços da Web

A especificação do Web Services Atomic Transaction (ou WS-AtomicTransaction) e a especificação do Web Services Coordination (ou WS-Coordination) definem protocolos para transações de curto prazo que permitem que sistemas de processamento de transações interoperem em um ambiente de serviços da web. Transações que usam WS-AtomicTransaction possuem as propriedades *ACID* de atomicidade, consistência, isolamento e durabilidade.

As especificações podem ser localizadas em OASIS.

Nota: O CICS suporta o nível de Novembro de 2004 das especificações.

Os aplicativos CICS que são implementados como provedores ou solicitantes de serviço da web podem participar em transações distribuídas com outras implementações de serviço da web que suportam as especificações.

Serviços de Registro

Os serviços de registro são a parte do modelo do WS-Coordination que permite que um aplicativo registre os protocolos de coordenação. Em uma transação

distribuída, os serviços de registro nos sistemas participantes se comunicam uns com os outros para permitir que os aplicativos conectados participem desses protocolos.

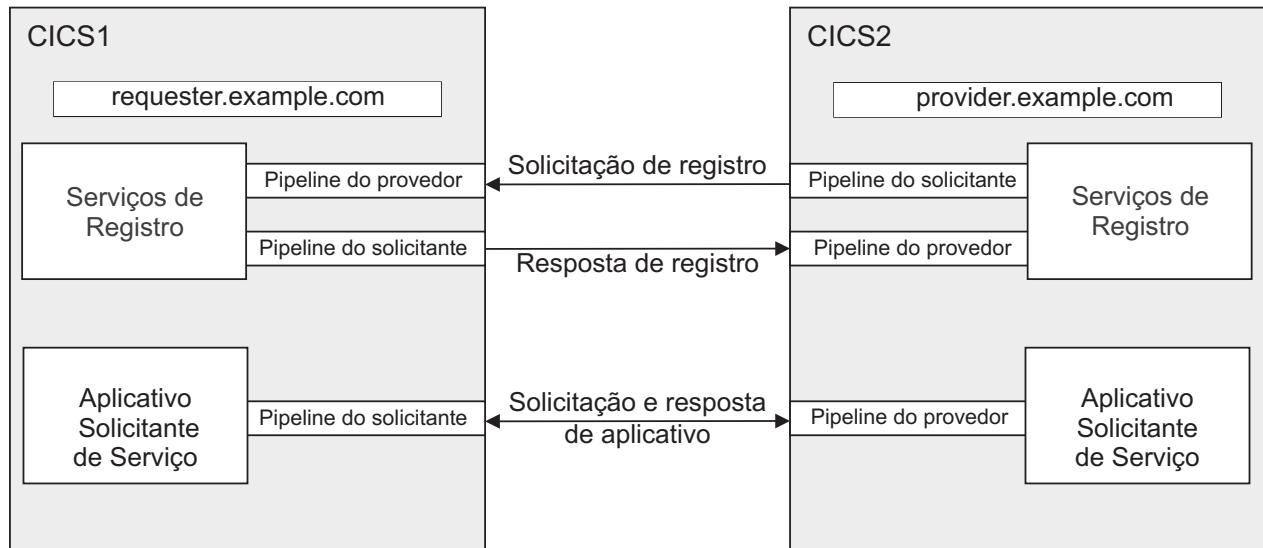


Figura 24. Serviços de Registro

Figura 24 mostra dois sistemas CICS, CICS1 e CICS2. Um aplicativo do solicitante de serviço no CICS1 chama um aplicativo do provedor de serviços no CICS2. As duas regiões do CICS e os aplicativos são configurados para que os dois aplicativos participem de uma única transação distribuída, usando os protocolos WS-Coordination. O aplicativo do solicitante de serviço é o coordenador e o aplicativo do provedor de serviços é o participante.

No suporte destes protocolos, os serviços de registro nas duas regiões do CICS interagem no início da transação e novamente durante o término da transação. Durante estas interações, os serviços de registro em ambas as regiões podem operar em momentos diferentes como um provedor de serviços e como um solicitante. Portanto, em cada região, os serviços de registro usam um pipeline do provedor de serviços e um pipeline do solicitante de serviço. Os pipelines são definidos no CICS com o PIPELINE e os recursos associados.

Os serviços de registro em cada região são associados a um endereço de terminal. Assim, no exemplo, serviços de registro no CICS1 possuem um endereço de terminal `requester.example.com`; no CICS2 possuem um endereço de terminal `provider.example.com`.

Em um CICSplex, é possível distribuir o pipeline do provedor de serviços de registro para uma região diferente. Isto é mostrado em Figura 25 na página 192.

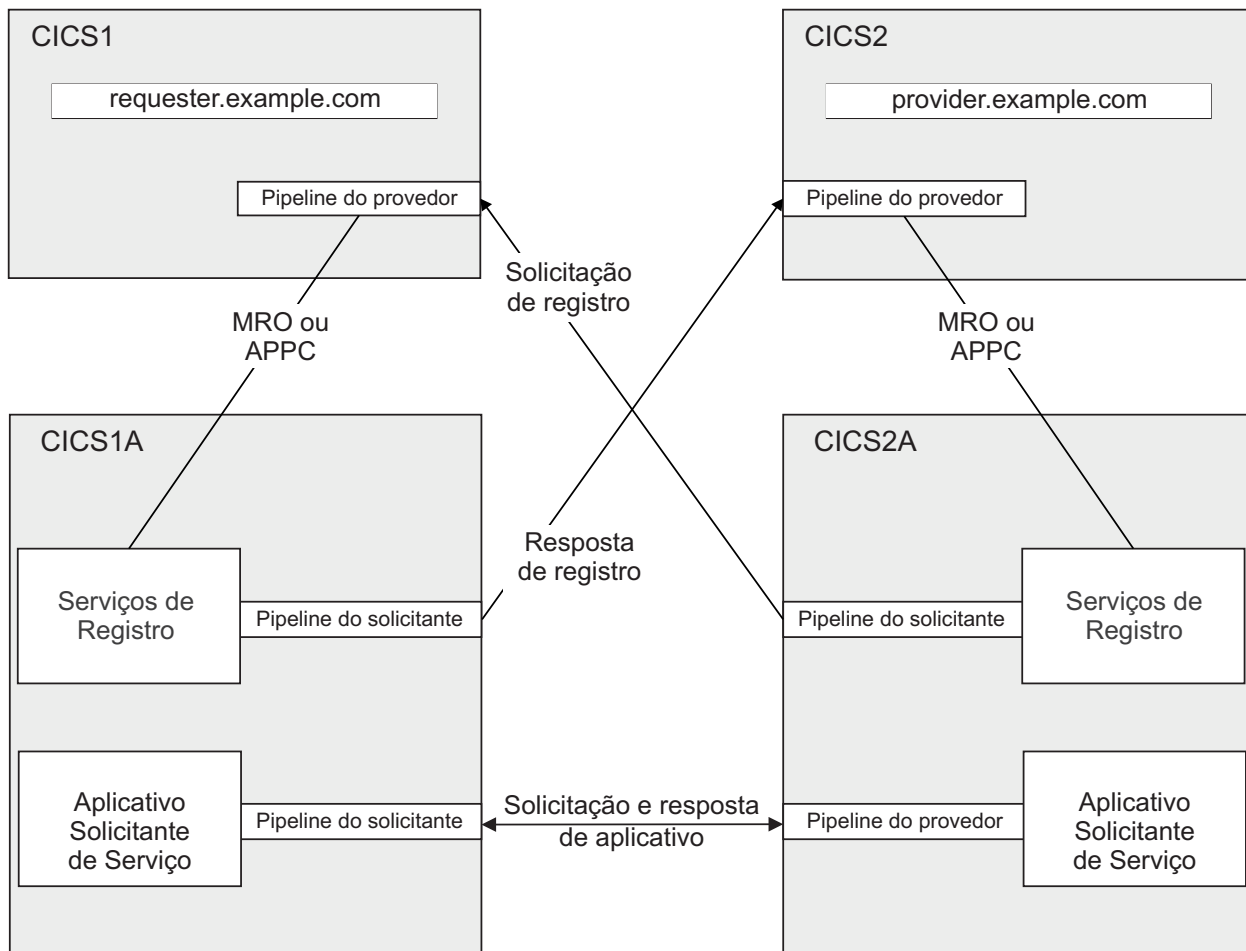


Figura 25. Serviços de Registro em um CICSplex

Nesta configuração, o pipeline do provedor se comunica com serviços de registro usando MRO ou APPC. O pipeline do solicitante de serviços de registro deve permanecer na mesma região que o pipeline do solicitante do aplicativo.

Esta configuração é útil quando seus aplicativos de solicitante e provedor de serviços são distribuídos em um grande número de regiões. Ao configurar pipelines do aplicativo para participarem de transações de serviço da web, você deve fornecer informações sobre o terminal de serviços de registro, fornecendo o endereço IP e o número da porta do pipeline do provedor de serviços de registro. Tendo um único terminal, é possível simplificar a configuração, porque todos os seus pipelines conterão as mesmas informações. Por exemplo, na Figura 25 o endereço IP especificado no pipeline do solicitante do aplicativo é requester.example.com.

Os mesmos argumentos se aplicam ao aplicativo do provedor de serviços. No exemplo, o pipeline do aplicativo do provedor especificará um endereço IP de requester.example.com.

Configurando o CICS para Transações de Serviço da Web

Para aplicativos do solicitante e provedor de serviço da web participarem de transações de serviço da web, você deve configurar o CICS de acordo instalando vários recursos do CICS.

Antes de Iniciar

Antes de poder instalar estes recursos, você deve conhecer o local dos arquivos de configuração de pipeline que o CICS fornece no suporte de transações de serviço da web. Por padrão, os arquivos de configuração são fornecidos no diretório `/usr/lpp/cicsts/cicsts54/pipeline/configs`, mas o caminho de arquivo padrão pode ter sido alterado durante a instalação do CICS.

Sobre Esta Tarefa

O suporte do CICS para transações de serviço da web usa um serviço de registro fornecido pelo CICS. Este serviço de registro consiste em um provedor de serviços e um solicitante de serviço. Você deve instalar recursos para o provedor de serviços e o solicitante de serviço; mesmo se seus aplicativos forem todos provedores de serviços ou todos solicitantes de serviços.

Você também deve instalar uma definição de programa para o programa do manipulador de cabeçalho que é chamado quando você executa seus aplicativos de provedor e solicitante de serviços.

Os recursos requeridos para configurar o CICS para transações de serviço da web são todos fornecidos no grupo DFHWSAT, exceto para DFHPIDIR que é fornecido em um dos grupos a seguir: DFHPIVS, DFHPIVR ou DFHPICF. O grupo DFHWSAT não é incluído na lista DFHLIST e, portanto, não é instalado automaticamente. Não é possível alterar os recursos fornecidos pelo CICS no grupo DFHWSAT.

Para configurar o CICS para transações de serviço da web:

Procedimento

1. Inclua o conjunto de dados DFHPIDIR em sua JCL de inicialização. DFHPIDIR armazena um mapeamento entre contextos e tarefas.
 - a. Inclua uma nova instrução DD para o conjunto de dados DFHPIDIR em sua JCL de inicialização do CICS
 - b. Crie o conjunto de dados DFHPIDIR usando informações em DFHDEFDS.JCL. O RECORDSIZE padrão de DFHPIDIR é 1 KB, que é adequado para a maioria das utilizações. É possível criar DFHPIDIR com um RECORDSIZE maior se necessário.
 - c. Instale o grupo apropriado para o conjunto de dados em seu sistema CICS: DFHPIVS, DFHPIVR ou DFHPICF. Para obter mais informações sobre esses grupos, consulte Definindo o conjunto de dados WS-AT.

Se desejar compartilhar o arquivo DFHPIDIR entre regiões do CICS, as regiões deverão ser conectadas logicamente por meio do MRO. Você deve instalar um conjunto de dados por grupo de regiões que estão agindo como um servidor lógico.

Dica: É recomendado que você não compartilhe conjuntos de dados entre regiões que não estão logicamente conectadas.

2. Copie o conteúdo do grupo DFHWSAT em um outro grupo. Não é possível alterar os recursos fornecidos pelo CICS no grupo DFHWSAT. No entanto, você deve alterar o atributo CONFIGFILE nos recursos PIPELINE.
3. Modifique o recurso PIPELINE do provedor do serviço de registro. O PIPELINE é nomeado DFHWSATP e especifica o arquivo de configuração de pipeline /usr/lpp/cicsts/cicsts54/pipeline/configs/registrationservicePROV.xml no atributo CONFIGFILE.
 - a. Altere o atributo CONFIGFILE para refletir o local do arquivo em seu sistema.
 - b. Deixe os outros atributos inalterados.

Use o arquivo de configuração de pipeline exatamente conforme fornecido; não altere seu conteúdo.

4. Instale o recurso PIPELINE. O recurso PIPELINE do provedor do serviço de registro não precisa estar na mesma região do CICS que seus aplicativos de provedor ou solicitante de serviço, mas deve estar conectado a essa região com uma conexão MRO ou APPC adequada.
5. Sem alterá-lo, instale o URIMAP que é usado pelo provedor de serviços de registro na mesma região que o PIPELINE. O URIMAP é nomeado DFHRSURI.
6. Modifique o recurso PIPELINE do solicitante do serviço de registro. O PIPELINE é nomeado DFHWSATR e especifica o arquivo de configuração de pipeline /usr/lpp/cicsts/cicsts54/pipeline/configs/registrationserviceREQ.xml no atributo CONFIGFILE.
 - a. Altere o atributo CONFIGFILE para refletir o local do arquivo em seu sistema.
 - b. Deixe os outros atributos inalterados.

Use o arquivo de configuração de pipeline exatamente conforme fornecido; não altere seu conteúdo.

7. Instale o recurso PIPELINE. O recurso PIPELINE do solicitante do serviço de registro deve estar na mesma região do CICS que os aplicativos de provedor ou solicitante de serviço.
8. Instale os programas usados pelo pipeline do provedor de serviço de registro na mesma região que seus recursos PIPELINE. Os programas são DFHWSATX, DFHWSATR e DFHPIRS. Se seus recursos PIPELINE estiverem em regiões diferentes, você deverá instalar estes programas em ambas as regiões.
9. Instale a definição de recurso PROGRAM para o programa do manipulador de cabeçalho. O programa é nomeado DFHWSATH. Instale o PROGRAM nas regiões nas quais seus aplicativos de provedor e solicitante de serviços são executados.

Resultados

O CICS agora é configurado para que seus aplicativos de solicitante e provedor de serviços possam participar de transações distribuídas usando protocolos WS-AtomicTransaction e WS-Coordination.

O que Fazer Depois

Agora você deve configurar cada aplicativo participante individualmente.

Configurando um Provedor de Serviços para Transações de Serviço da Web

Se um aplicativo do provedor de serviços precisar participar de transações de serviço da web, o arquivo de configuração de pipeline deverá especificar um elemento `<headerprogram>` e um elemento `<service_parameter_list>`.

Antes de Iniciar

Se desejar que seu aplicativo do provedor de serviços participe das transações de serviço da web, ele deverá usar protocolos SOAP para se comunicar com o solicitante de serviço e você deverá configurar seu pipeline para usar um dos manipuladores de mensagem SOAP fornecidos pelo CICS. Mesmo se você tiver configurado seu aplicativo do provedor de serviços corretamente, ele participará das transações de serviço da web com o solicitante de serviço somente se o aplicativo do solicitante tiver sido configurado para participar.

Sobre Esta Tarefa

Além das informações de configuração de pipeline que são específicas para seu aplicativo, o arquivo de configuração deve conter informações que o CICS usa para assegurar que seu aplicativo participe das transações de serviço da web.

O CICS fornece um exemplo de um arquivo de configuração de pipeline contendo estas informações no diretório do arquivo `/usr/lpp/cicsts/cicsts54/samples/pipelines/wsatprovider.xml` (em que `/usr/lpp/cicsts/cicsts54` é o diretório de instalação padrão para arquivos CICS no z/OS UNIX).

Procedimento

1. Na definição de seu manipulador de terminal, codifique um elemento `<headerprogram>` no elemento `<cics_soap_1.1_handler>`, `<cics_soap_1.2_handler>`, `<cics_soap_1.1_handler_java>` ou `<cics_soap_1.2_handler_java>`. Codifique os elementos `<program_name>`, `<namespace>`, `<localname>` e `<mandatory>` exatamente conforme mostrado neste exemplo:

```
<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>
```

Inclua outros elementos `<headerprogram>` se seu aplicativo precisar deles.

2. Codifique um elemento `<registration_service_endpoint>` em um `<service_parameter_list>`. Codifique o `<registration_service_endpoint>` conforme a seguir:

```
<registration_service_endpoint>
http://address:port/cicswsat/RegistrationService
</registration_service_endpoint>
```

address é o endereço IP da região do CICS na qual o pipeline do provedor de serviço de registro está localizado.

port é o número da porta usado pelo pipeline do provedor do serviço de registro.

Codifique tudo mais exatamente conforme mostrado; a sequência cicswsat/RegistrationService corresponde ao atributo PATH de URIMAP DFHRSURI:

```
<registration_service_endpoint>  
http://provider.example.com:7160/cicswsat/RegistrationService  
</registration_service_endpoint>
```

Configurando um Solicitante de Serviço para Transações de Serviço da Web

Se um aplicativo solicitante de serviço precisar participar de transações de serviço da web, o arquivo de configuração de pipeline especificará um elemento <headerprogram> e um elemento <service_parameter_list>.

Antes de Iniciar

Se desejar que o aplicativo do solicitante de serviço participe de transações de serviço da web, ele deverá usar protocolos SOAP para se comunicar com o provedor de serviços e seu pipeline deverá ser configurado para usar um dos manipuladores de mensagem SOAP fornecidos pelo CICS. Mesmo se você tiver configurado seu aplicativo do solicitante de serviço corretamente, ele participará somente das transações de serviço da web com o provedor de serviços se o aplicativo do provedor tiver sido configurado para participar.

Sobre Esta Tarefa

Além das informações de configuração de pipeline que são específicas para seu aplicativo, o arquivo de configuração deve conter informações que o CICS usa para assegurar que seu aplicativo participe das transações de serviço da web.

O CICS fornece um exemplo de um arquivo de configuração de pipeline contendo estas informações no diretório do arquivo /usr/lpp/cicsts/cicsts54/samples/pipelines/wsatre requester.xml (em que /usr/lpp/cicsts/cicsts54 é o diretório de instalação padrão para arquivos CICS no z/OS UNIX).

Procedimento

1. Codifique um elemento <headerprogram> no elemento <cics_soap_1.1_handler>, <cics_soap_1.2_handler>, <cics_soap_1.1_handler_java> ou <cics_soap_1.2_handler_java>. Codifique os elementos <program_name>, <namespace>, <localname> e <mandatory> exatamente como mostrado no exemplo a seguir:

```
<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>
```

É possível incluir outros elementos <headerprogram> se seu aplicativo precisar deles.

2. Codifique um elemento <registration_service_endpoint> em um <service_parameter_list>. Codifique o <registration_service_endpoint> conforme a seguir:

```
<registration_service_endpoint>  
http://address:port/cicswsat/RegistrationService  
</registration_service_endpoint>
```

address é o endereço IP da região do CICS na qual o pipeline do provedor de serviço de registro está localizado.

port é o número da porta usado pelo pipeline do provedor do serviço de registro.

Não deve haver espaço entre o início do elemento

<registration_service_endpoint>, seu conteúdo e o final do elemento

<registration_service_endpoint>. Os espaços foram incluídos neste exemplo para clareza.

3. Se desejar que o CICS crie um novo contexto transacional para cada solicitação, em vez de usar o mesmo para solicitações na mesma unidade de trabalho, inclua o elemento vazio, <new_tx_context_required/>, em um <service_parameter_list> em seu arquivo de configuração de pipeline:

```
<service_parameter_list>
  <registration_service_endpoint>
    http://requester.example.com:7159/cicswsat/RegistrationService
  </registration_service_endpoint>
  <new_tx_context_required/>
</service_parameter_list>
```

Não deve haver espaço entre o início do elemento

<registration_service_endpoint>, seu conteúdo e o final do elemento

<registration_service_endpoint>. Os espaços foram incluídos neste exemplo para clareza.

A configuração de <new_tx_context_required/> não é o padrão para o CICS e não é incluída no arquivo de configuração de pipeline de exemplo, wsatprovider.xml. Se você incluir <new_tx_context_required/> em um <service_parameter_list> para seu arquivo de configuração de pipeline, chamadas de loopback para o CICS serão permitidas, portanto, esteja ciente que pode ocorrer um conflito nesta situação.

Determinando se a Mensagem SOAP faz Parte de uma Transação Atômica

Quando um serviço da web do CICS é chamado no pipeline da transação atômica, a mensagem SOAP não precisa necessariamente fazer parte de uma transação atômica.

Sobre Esta Tarefa

O elemento <soapenv:Header> contém informações específicas quando a mensagem SOAP faz parte de uma transação atômica. Para descobrir se a mensagem SOAP faz parte de uma transação atômica, é possível:

Procedimento

- Consultar o conteúdo do elemento <soapenv:Header> usando um rastreo.
 1. Execute um rastreo auxiliar usando o componente PI e configure o nível de rastreo como 2.
 2. Procure o ponto de rastreo PI 0A31, que contém as informações para o contêiner de solicitação. Em específico, procure PIIS EVENT - REQUEST_CNT que aparece logo antes do elemento <cicswsa:Action>.
- Use um programa do manipulador de mensagem gravado pelo usuário no pipeline DFHWSATP para exibir o conteúdo do contêiner DFHREQUEST quando ele contém os dados RECEIVE-REQUEST. Se você optar por esta abordagem, certifique-se de definir o programa do manipulador de mensagem no arquivo de configuração de pipeline.

Exemplo

O exemplo a seguir mostra as informações que você poderia ver no cabeçalho do envelope SOAP para uma transação atômica.

```
<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. O CoordinationContext indica que a mensagem SOAP destina-se a participar de uma transação atômica. Ele contém as informações necessárias para que o provedor de serviço da web faça parte do serviço de coordenação, assumindo que o provedor está configurado para reconhecer e processar o cabeçalho.
2. O CoordinationType indica a versão da especificação WS-AT com a qual o contexto de coordenação está em conformidade.
3. A coordenação RegistrationService descreve onde o ponto de registro do coordenador está e as informações que o serviço da web participante deve retornar ao coordenador quando ele tenta registrar-se como um componente da transação atômica.

Verificando o Progresso de uma Transação Atômica

Quando um serviço da web do CICS é chamado como parte de uma transação atômica, a transação passa por vários estados. Estes estados indicam se a transação foi bem-sucedida ou precisou ser recuperada.

Sobre Esta Tarefa

Se precisar acessar estas informações, é possível:

Procedimento

- Consultar o conteúdo do elemento <cicswsa:Action> usando um rastreio.
 1. Execute um rastreio auxiliar usando o componente PI e configure o nível de rastreio como 2.
 2. Procure o ponto de rastreio PI 0A31, que contém as informações para o contêiner de solicitação. Em específico, procure PIIS EVENT - REQUEST_CNT que aparece logo antes do elemento <cicswsa:Action>.

- Use um programa do manipulador de mensagem gravado pelo usuário nos pipelines DFHWSATR e DFHWSATP para exibir o conteúdo de contêineres DFHWS-SOAPACTION. Se você optar por esta abordagem, certifique-se de definir o programa do manipulador de mensagem nos arquivos de configuração de pipeline.

Exemplo

Os estados para uma transação que é concluída com sucesso e é confirmada são:

```
"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 "
```

Os estados para uma transação que é recuperada são:

```
"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"
```

Suporte para Otimização de MTOM/XOP de Dados Binários

Nas mensagens SOAP padrão, os objetos binários são codificados em base64 e incluídos no corpo da mensagem, o que aumenta seu tamanho em 33%. Para objetos binários muito grandes, este aumento de tamanho pode impactar significativamente o tempo de transmissão. A implementação de MTOM/XOP oferece uma solução para este problema.

As especificações SOAP Message Transmission Optimization Mechanism (MTOM) e XML-binary Optimized Packaging (XOP), geralmente referidas como MTOM/XOP, definem um método para otimizar a transmissão de objetos de dados base64Binary grandes dentro de mensagens SOAP.

- A especificação MTOM define conceitualmente um método para otimizar mensagens SOAP separando dados binários, que de outra forma seriam codificados em base64, e enviando-os em anexos binários separados usando uma mensagem Multiparte/Relacionada MIME. Este tipo de mensagem MIME é chamado de *mensagem MTOM*. O envio de dados em formato binário reduz significativamente seu tamanho, otimizando, portanto, a transmissão da mensagem SOAP.
- A especificação XOP define uma implementação para otimizar mensagens XML utilizando anexos binários em um formato de pacote que inclui mas não está limitado a mensagens MIME.

O CICS implementa o suporte para estas especificações em pipelines do solicitante e do provedor quando o protocolo de transporte é WebSphere MQ, HTTP ou HTTPS. Como uma alternativa para incluir os dados base64Binary diretamente na mensagem SOAP, os aplicativos CICS que são implementados como provedores ou solicitantes de serviço da web podem usar este suporte para enviar e receber mensagens MTOM com anexos binários.

Você pode configurar este suporte utilizando opções adicionais no arquivo de configuração de enfileiramento.

MTOM/XOP e SOAP

Quando MTOM/XOP é usado para otimizar uma mensagem SOAP, ele é serializado em uma mensagem MIME Multiparte/Relacionada usando processamento de XOP. Os dados base64Binary são extraídos da mensagem SOAP e empacotados como anexos binários separados dentro da mensagem MIME, de uma maneira semelhante aos anexos de e-mail.

O tamanho dos dados base64Binary é significativamente reduzido porque os anexos são codificados em formato binário. O XML na mensagem SOAP é, então, convertido no formato XOP, substituindo os dados base64Binary por um elemento `<xop:Include>` especial que referencia o anexo MIME relevante utilizando um URI.

A mensagem SOAP modificada é chamada de *documento XOP* e forma o documento raiz dentro da mensagem. O documento XOP e os anexos binários juntos formam o *pacote XOP*. Quando aplicado à especificação SOAP MTOM, o pacote XOP é uma mensagem MIME no formato MTOM.

O documento raiz é identificado referenciando seu Content-ID no cabeçalho content-type geral da mensagem MIME. A seguir há um exemplo de um cabeçalho content-type:

```
Content-Type: Multipart/Related; boundary=MIME_boundary;  
type="application/soap+xml"; start="<claim@insurance.com>"
```

O parâmetro **start** contém o Content-ID do documento XOP. Se este parâmetro não for incluído no cabeçalho content-type, a primeira parte na mensagem MIME será assumida como sendo o documento XOP.

A ordem dos anexos na mensagem MIME não é importante. Em algumas mensagens, por exemplo, os anexos binários poderão aparecer antes do documento XOP. Um aplicativo que manipula mensagens MIME não deve depender dos anexos que aparecem em uma ordem específica. Para obter informações detalhadas, leia as especificações MTOM/XOP.

O exemplo a seguir demonstra como uma mensagem SOAP simples que contém uma imagem JPEG é otimizada usando processamento de XOP. A mensagem SOAP é conforme a seguir:

```
<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>
```

Uma versão de MTOM/XOP desta mensagem SOAP é conforme a seguir:

```
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. O parâmetro **start** indica qual parte da mensagem MIME é o documento XOP raiz.
2. O valor de Content-ID identifica uma parte da mensagem MIME. Neste caso, ele é o documento XOP raiz.
3. O elemento <xop:Include> referencia o anexo binário JPEG.
4. O Content-ID identifica o JPEG no anexo binário.

Mensagens MTOM e Anexos Binários no CICS

O CICS suporta e controla a manipulação de mensagens do MTOM tanto no provedor de serviços da web como nos pipelines do solicitante utilizando o programa do manipulador do MTOM e o processamento de XOP.

É possível configurar e ativar o suporte a MTOM utilizando o arquivo de configuração de pipeline. Se o suporte a MTOM for ativado para um pipeline, o CICS desempacotará as mensagens do MTOM de entrada automaticamente e empacotará as mensagens de saída. Se o suporte a MTOM não estiver ativado para um pipeline e o CICS receber uma mensagem do MTOM, os pipelines baseados em Java aceitarão a mensagem do MTOM de entrada, no entanto, outros pipelines SOAP rejeitarão a mensagem do MTOM de entrada com uma falha de SOAP.

Opções de Configuração para Pipelines Baseados em Java

É possível configurar um pipeline do provedor para executar as tarefas a seguir:

- Aceitar mensagens do MTOM, mas jamais enviar mensagens de resposta do MTOM.
- Aceitar mensagens do MTOM e sempre enviar mensagens de resposta do MTOM.
- Processar documentos XOP e anexos binários no modo Axis2.

É possível configurar um pipeline do solicitante para executar as tarefas a seguir:

- Jamais enviar uma mensagem do MTOM, mas aceitar mensagens de resposta do MTOM.
- Sempre enviar mensagens do MTOM e aceitar mensagens de resposta do MTOM.
- Processar documentos XOP e anexos binários no modo Axis2.

Opções de Configuração para Pipelines que Não Suportam Java

É possível configurar um pipeline do provedor para executar as tarefas a seguir:

- Aceitar mensagens do MTOM, mas jamais enviar mensagens de resposta do MTOM.

- Aceitar mensagens do MTOM e enviar o mesmo tipo de mensagem de resposta.
- Aceitar mensagens do MTOM, mas enviar somente mensagens do MTOM quando houver anexos binários presentes.
- Aceitar mensagens do MTOM e sempre enviar mensagens de resposta do MTOM.
- Processar documentos XOP e anexos binários no modo direto ou de compatibilidade.

É possível configurar um pipeline do solicitante para executar as tarefas a seguir:

- Jamais enviar uma mensagem do MTOM, mas aceitar mensagens de resposta do MTOM.
- Enviar somente mensagens do MTOM quando houver anexos binários e aceitar mensagens de resposta do MTOM.
- Sempre enviar mensagens do MTOM e aceitar mensagens de resposta do MTOM.
- Processar documentos XOP e anexos binários no modo direto ou de compatibilidade.

Modos de suporte

Há três modos de suporte fornecidos no pipeline para manipular documentos XOP e quaisquer anexos binários associados.

Modo Axis2

O modo Axis2 é usado quando o manipulador de terminal de seu pipeline de serviços da web é o manipulador de mensagem `<cics_soap_1.1_handler_java>` ou `<cics_soap_1.2_handler_java>`.

Modo Direto

No modo direto, os anexos binários associados a uma mensagem do MTOM de entrada ou de saída são transmitidos em contêineres por meio do enfileiramento e manipulados diretamente pelo aplicativo, sem a necessidade de realizar nenhuma conversão de dados.

Modo de Compatibilidade

O modo de compatibilidade é utilizado quando o processamento enfileirado requer que a mensagem esteja em um formato XML padrão, com qualquer dado binário armazenado como campos `base64Binary` na mensagem. Para mensagens de entrada, o documento XOP e os anexos binários são reconstituídos em uma mensagem XML padrão, no início do pipeline quando o Web Services Security está ativado, ou no final do pipeline quando a validação do serviço da web está ativada. Para mensagens de saída, uma mensagem XML padrão é criada e transmitida pelo enfileiramento. Esta mensagem XML é convertida para o formato XOP pelo manipulador MTOM logo antes do CICS enviá-la.

O modo de compatibilidade é muito menos eficiente do que o modo direto, pois os dados binários são convertidos no formato base64 e desconvertidos. Entretanto, ele permite que seus serviços da web interoperem com outros solicitantes e provedores de serviços da web do MTOM/XOP sem a necessidade de alterar seus aplicativos.

Processamento de Mensagens MTOM de Entrada para Pipelines que Não Suportam Java:

Quando o manipulador MTOM é ativado nos pipelines que não suportam Java, ele verifica os cabeçalhos da mensagem de entrada no contêiner DFHREQUEST ou DFHRESPONSE para determinar o formato da mensagem durante o processamento de manipulação de transporte.

Quando uma mensagem MIME Multiparte/Relacionada é recebida, o manipulador MTOM descompacta a mensagem conforme a seguir:

1. Ele coloca os cabeçalhos e dados binários de cada anexo binário em contêineres separados.
2. Ele coloca a lista de contêineres no contêiner DFHWS-XOP-IN.
3. Ele coloca o documento XOP, que formou a raiz da mensagem, de volta no contêiner DFHREQUEST ou DFHRESPONSE, substituindo a mensagem original.

Se não houver anexos binários, o documento XOP é manipulado como uma mensagem XML normal e nenhum processamento de XOP é necessário. Se houver quaisquer anexos binários, o processamento de XOP será ativado para a mensagem.

Se o processamento de XOP for ativado, o manipulador MTOM verificará as propriedades de pipeline para determinar se a mensagem atual deve ser processada no modo direto ou de compatibilidade e coloca estas informações no contêiner DFHWS-MTOM-IN.

No modo de provedor, o manipulador MTOM também cria o contêiner DFHWS-MTOM-OUT para determinar como a mensagem de resposta de saída deve ser processada.

Modo Direto

Quando o suporte de serviços da web do CICS está sendo usado, ou seja, quando um pipeline do provedor de serviços usa o manipulador de aplicativo DFHPITP ou um pipeline do solicitante de serviço é chamado usando **INVOKE WEBSERVICE**, o pipeline pode processar o documento XOP e anexos binários no modo direto.

Neste modo, o documento XOP e os contêineres associados são transmitidos pelo manipulador MTOM para o próximo manipulador de mensagem no pipeline para processamento. O suporte de serviços da web do CICS interpreta os elementos <xop:Include>. Se o campo base64Binary for representado como um contêiner na estrutura de dados do aplicativo, o nome do contêiner do anexo será armazenado na estrutura. Se o campo for representado como uma sequência de comprimento variável ou fixo, o conteúdo do contêiner será copiado no campo da estrutura de dados do aplicativo relevante. A estrutura de dados é, então, transmitida ao programa de aplicativo.

Modo de Compatibilidade

Se seu pipeline for configurado para usar um manipulador de aplicativo customizado, ou o Web Services Security também estiver ativado, a mensagem será processada no modo de compatibilidade. Neste modo, o documento XOP e os anexos binários são imediatamente reconstituídos em uma mensagem SOAP

usando processamento de XOP, de forma que o conteúdo possa ser processado com sucesso no pipeline. O processamento de XOP executa as tarefas a seguir:

1. Varre o documento XOP para elementos <xop:Include>, substituindo cada ocorrência pelos dados binários do anexo referenciado no formato codificado em base64.
2. Descarta o contêiner DFHWS-XOP-IN e todos os contêineres do anexo.

A mensagem SOAP reconstituída é, então, transmitida ao próximo manipulador no pipeline para ser processada normalmente.

Se a validação de serviço da web estiver ativada, o pipeline alternará para o modo de compatibilidade quando a mensagem atingir o manipulador de aplicativo. A mensagem é reconstituída em uma mensagem SOAP, validada e transmitida ao aplicativo.

Processamento de Mensagens MTOM de Saída para Pipelines que Não Suportam Java:

Quando um pipeline que não suporta Java está configurado para enviar mensagens MTOM de saída, o serviço da web e propriedades do pipeline são verificados para determinar como a mensagem deve ser processada e enviada.

Estas propriedades são armazenadas em dois contêineres, DFHWS-MTOM-OUT e DFHWS-XOP-OUT. Em um pipeline no modo do solicitante, esses contêineres são criados pelo CICS quando o aplicativo emite o comando **EXEC CICS INVOKE WEBSERVICE**. Em um pipeline no modo de provedor, o contêiner DFHWS-MTOM-OUT já está inicializado com as opções que foram determinadas quando a mensagem de entrada foi recebida.

Se a mensagem de saída pode ser processada no modo direto, a otimização da mensagem ocorre imediatamente. Se a mensagem de saída precisa ser processada no modo de compatibilidade, a otimização ocorre no final do processamento de pipeline.

Se você não tiver implementado seu aplicativo do provedor ou do solicitante de serviço da web usando o assistente de serviços da web do CICS, ou se você tiver a validação de serviço da web ativada ou o Web Services Security ativado em seu pipeline, a mensagem de saída será processada no modo de compatibilidade.

Modo Direto

No modo direto, o seguinte processamento ocorre:

1. Um documento XOP é construído a partir de estrutura de dados do aplicativo no contêiner DFHWS-DATA. Qualquer campo binário que seja igual ou maior em tamanho do que 1500 bytes seja identificado e os dados binários e cabeçalhos MIME que descrevem o anexo binário serão colocados em contêineres separados. Se os dados binários já estiverem em um contêiner, esse contêiner será usado diretamente como o anexo. Um elemento <xop:Include> é, então, inserido no XML no lugar dos dados binários codificados em base64 usuais usando um Content-ID gerado. Por exemplo:

```
<xop:Include href="cid:generated-content-ID-value"
  xmlns:xop="http://www.w3.org/2004/08/xop/include">
```
2. Todos os contêineres são incluídos na lista de anexos no contêiner DFHWS-XOP-OUT.

3. Quando o manipulador SOAP tiver processado DFHWS-DATA, o documento XOP e o envelope SOAP serão armazenados no contêiner DFHREQUEST ou DFHRESPONSE e processados por meio do pipeline.
4. Quando o último manipulador de mensagem tiver concluído, o manipulador do MTOM empacotará o documento XOP e os anexos binários em uma mensagem MIME Multiparte/Relacionada e o enviará para o solicitante ou provedor de serviços da web. O contêiner DFHWS-XOP-OUT e quaisquer contêineres associados são, então, descartados.

Modo de Compatibilidade

Se o pipeline não for capaz de manipular o documento XOP diretamente, o processamento a seguir ocorrerá:

1. O corpo SOAP é construído no DFHWS-DATA a partir da estrutura de dados do aplicativo e processado no enfileiramento normalmente.
2. Quando o manipulador final tiver concluído o processamento da mensagem, o manipulador MTOM verificará as opções no contêiner DFHWS-MTOM-OUT para determinar se o MTOM deve ser usado, opcionalmente, levando em conta se quaisquer anexos binários estão presentes. Se o manipulador MTOM determinar que o MTOM não é necessário, nenhum processamento de XOP ocorrerá e uma mensagem SOAP será enviada pelo CICS normalmente.
3. Se o manipulador do MTOM determinar que a mensagem de saída deve ser enviada no formato MTOM, o processamento de XOP varrerá a mensagem em busca de campos elegíveis para dividir os dados em anexos binários. Para que um campo seja elegível, ele deverá ter o atributo **contentType** MIME especificado no elemento e o valor binário associado deve consistir em dados base64Binary válidos no formato canônico. O tamanho dos dados deve ser maior que 1500 bytes. O processamento de XOP cria os anexos binários e a lista de anexos e, em seguida, substitui os campos pelos elementos <xop:Include>.
4. O manipulador MTOM empacota o documento XOP e os anexos binários como uma mensagem MIME Multiparte/Relacionada e o CICS a envia ao solicitante ou provedor de serviço da web.

Restrições ao Usar MTOM/XOP

Para suportar MTOM/XOP, é possível especificar o elemento <mtom> em seu arquivo de configuração de pipeline ou ativar o manipulador de MTOM em seu pipeline. No entanto, há restrições associadas a cada método.

Restrições para Pipelines Baseados em Java:

A especificação do elemento <mtom> no arquivo de configuração de pipeline ativa o suporte de MTOM/XOP para seu pipeline baseado em Java. No entanto, há restrições com esta implementação de MTOM/XOP.

Manipulador de Aplicativo DFHPITP

O modo Axis2 de suporte de MTOM/XOP não pode ser usado com pipelines que especificam DFHPITP como o manipulador de aplicativo.

Segurança WS

O modo Axis2 de suporte de MTOM/XOP não pode ser usado com pipelines que usam configurações de WS-Security que requerem assinaturas XML.

Usando o comando INQUIRE PIPELINE

Se um comando **INQUIRE PIPELINE** for emitido em um pipeline baseado em Java usando o modo Axis2 de suporte de MTOM/XOP, os atributos

Mtomst, **Sendmtomst**, **Mtomnoxopst**, **Xopsupportst** e **Xopdirectst** serão relatados como Nomtom. Para obter mais informações, consulte INQUIRE PIPELINE.

Restrições para Outros Pipelines SOAP:

A ativação do manipulador MTOM no pipeline significa que você pode suportar implementações de serviço da web que usam a otimização de MTOM/XOP. A opção de modo de compatibilidade significa que você pode interoperar com estes serviços da web sem precisar alterar seus aplicativos de serviço da web. No entanto, há determinadas situações em que não é possível usar MTOM/XOP ou seu uso é restrito.

Usando o assistente de serviços da web do CICS

A otimização de modo direto para MTOM/XOP está disponível somente se você está usando DFHWS2LS em um nível de mapeamento de pelo menos 1.2 e o documento WSDL contém pelo menos um campo de tipo xsd:base64Binary. Os serviços da web que são ativados usando DFHLS2WS não são elegíveis para otimização de XOP.

Os serviços da web gerados usando DFHLS2WS com CHAR-VARYING=BINARY especificado podem ser elegíveis para as otimizações de MTOM/XOP. Outros serviços da web gerados usando DFHLS2WS não contêm dados binários e não são elegíveis para as otimizações de MTOM/XOP, mas funcionarão normalmente em um PIPELINE que suporta MTOM/XOP.

Pipelines do Provedor

O CICS fornece um manipulador de aplicativo padrão chamado DFHPITP que pode ser configurado em um pipeline do provedor. Este manipulador de aplicativo é capaz de manipular documentos XOP e criar os contêineres necessários para suportar o processamento de pipeline no modo direto e de compatibilidade. Se estiver usando seu próprio manipulador de aplicativo em um pipeline do provedor e desejar ativar o MTOM/XOP, você deverá configurar o pipeline para executar no modo de compatibilidade.

Pipelines do Solicitante

Se seus aplicativos usarem o comando **INVOKE WEBSERVICE**, o CICS manipulará a otimização da mensagem SOAP para você no modo direto e de compatibilidade. Se estiver usando o programa DFHPIRT para iniciar o pipeline, será possível somente enviar e receber mensagens MIME Multipartes/Relacionadas no modo de compatibilidade.

Segurança de serviços da web

Se ativar o manipulador MTOM no arquivo de configuração de pipeline para executar no modo direto, e você também ativar o manipulador de mensagem Web Services Security, o pipeline suportará somente a manipulação de mensagens MTOM no modo de compatibilidade.

Manipulando Dados Binários

Quando tiver dados binários grandes para incluir em seu serviço da web, por exemplo um arquivo gráfico tal como um JPEG, será possível usar MTOM/XOP para otimizar o tamanho da mensagem que é enviada ao provedor ou solicitante de serviços. O tamanho mínimo de dados binários que podem ser otimizados usando MTOM/XOP é 1500 bytes. Se os dados binários em um campo forem menores do que 1500 bytes, o CICS não otimizará o campo.

Conforme indicado na especificação de XOP, não deve haver espaço em branco nos dados base64Binary. Quaisquer programas de aplicativo que produzem dados base64Binary devem usar o formulário canônico. Se os dados base64Binary em uma mensagem de saída contiverem espaço em branco, o CICS não converterá os dados em um anexo binário. Quando dados base64Binary são gerados pelo CICS, os campos são fornecidos no formulário canônico e, portanto, não contêm espaço em branco.

O atributo **contentType** deve estar presente em campos base64Binary para que o processamento de XOP ocorra no modo de compatibilidade em mensagens de saída. O atributo **contentType** não deve estar presente em campos hexBinary.

Validação do Serviço da Web

Se você ativar a validação de serviço da web, ocorrerá o processamento de pipeline a seguir:

- Se um documento XOP de entrada tiver sido transmitido por meio do pipeline no modo direto, o CICS alternará automaticamente para o modo de compatibilidade e o converterá de volta para o XML padrão quando o suporte de serviço da web do CICS estiver prestes a validar o documento.
- Uma mensagem SOAP de saída é gerada como XML padrão e é processada no modo de compatibilidade.

O processamento de pipeline extra é necessário porque o processamento da validação não pode manipular o conteúdo de documentos XOP.

Configurando o CICS para Suportar MTOM/XOP

Para suportar mensagens MTOM no CICS, você deve especificar o suporte de MTOM/XOP correto para seu tipo de pipeline em seus arquivos de configuração de pipeline.

Configurando o Suporte a MTOM/XOP para Pipelines Baseados em Java:

Para configurar o suporte a MTOM/XOP para pipelines baseados em Java, você deve incluir o elemento <mtom> em seus arquivos de configuração de pipeline.

Antes de Iniciar

Antes de executar esta tarefa, você deve identificar ou criar os arquivos de configuração de pipeline nos quais incluirá informações de configuração para MTOM/XOP.

Sobre Esta Tarefa

Se o elemento <mtom> for definido em seu arquivo de configuração de pipeline, o suporte a MTOM será ativado para todas as mensagens de entrada e saída. Entretanto, se este elemento não for especificado no arquivo de configuração de pipeline, o suporte de MTOM será ativado apenas para mensagens de entrada.

Procedimento

Inclua um elemento <mtom> em seu arquivo de configuração de pipeline. Este elemento deve ser definido após o elemento <addressing> opcional e antes do elemento <headerprogram> opcional.

Exemplo

Para um pipeline do modo de provedor ou solicitante, você poderá 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>
```

Configurando MTOM/XOP para Outros Pipelines SOAP:

Para configurar o suporte de MTOM/XOP para pipelines que não usam os manipuladores <cics_soap_1.1_handler_java> ou <cics_soap_1.2_handler_java>, você deve incluir o manipulador MTOM em seus arquivos de configuração de pipeline.

Antes de Iniciar

Antes de executar esta tarefa, você deve identificar ou criar os arquivos de configuração de pipeline nos quais incluirá informações de configuração para MTOM/XOP.

Procedimento

1. Inclua um elemento <cics_mtom_handler> em seu arquivo de configuração de pipeline. Este elemento deve ser o primeiro no elemento <provider_pipeline> e o último elemento antes de <service_parameter_list> no elemento <requester_pipeline>. Codifique os elementos a seguir:

```
<cics_mtom_handler>
  <dfhmtom_configuration version="1">
  </dfhmtom_configuration>
</cics_mtom_handler>
```

O elemento <dfhmtom_configuration> é um contêiner para os outros elementos na configuração. Se desejar aceitar as configurações padrão para processamento de MTOM/XOP, é possível especificar um elemento vazio conforme a seguir:

```
<cics_mtom_handler/>
```

2. Opcional: Codifique um elemento <mtom_options>. Em um pipeline de provedor de serviços e do solicitante de serviço, este elemento especifica se a mensagem de saída deve ser empacotada como uma mensagem MTOM.
 - a. Codifique o atributo **send_mtom** para definir se a mensagem de saída deve ser enviada como uma mensagem MTOM. Para obter detalhes deste atributo, consulte “O Elemento <mtom_options>” na página 115.
 - b. Codifique o atributo **send_when_no_xop** para definir se a mensagem de saída deve ser enviada como uma mensagem MTOM quando não há anexos binários presentes. Para obter detalhes deste atributo, consulte “O Elemento <mtom_options>” na página 115.
3. Opcional: Codifique um elemento <xop_options> com um atributo **apphandler_supports_xop**. Isto especifica se o manipulador de aplicativo é capaz de manipular documentos XOP diretamente. Se você não incluir este atributo, o padrão dependerá de se o elemento <apphandler> especifica

DFHPITP ou um outro programa. Para obter detalhes deste atributo, consulte “O Elemento <xop_options>” na página 116.

4. Opcional: Codifique um elemento <mime_options> com um atributo **content_id_domain**. Isto especifica o nome de domínio que deve ser usado ao gerar valores de content-ID MIME, que são usados para identificar anexos binários. Para obter detalhes deste atributo, consulte “O Elemento <mime_options>” na página 118.

Exemplo

O exemplo a seguir mostra um elemento <cics_mtom_handler> concluído no qual todos os elementos opcionais estão presentes:

```
<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>
```

Suporte para Web Services Addressing

O CICS suporta serviços que usam as especificações Worldwide Web Consortium (W3C) Web Services Addressing (WS-Addressing). Esta família de especificações fornece mecanismos independentes de transporte para endereçar serviços da web e facilitar o endereçamento de ponta a ponta.

O CICS assegura que seus aplicativos de serviço da web existentes possam aceitar solicitações de serviços da web que usam o WS-Addressing. Também é possível criar novos serviços da web que usam referências de terminal e propriedades de endereçamento de mensagem nas mensagens SOAP.

O WS-Addressing inclui informações de endereçamento, no formato de Message Addressing Properties (MAPs), para cabeçalhos de mensagem SOAP. Os MAPs incluem informações de sistemas de mensagens, como um ID de mensagem exclusivo e referências de terminal que detalham a origem da mensagem, o destino da mensagem e o destino das mensagens de resposta ou com falhas. Uma referência de terminal (EPR) é um tipo específico de MAP, que inclui o endereço de destino da mensagem, os parâmetros de referência opcionais para uso do aplicativo e metadados opcionais.

Recursos de Suporte ao WS-Addressing

O CICS inclui os seguintes recursos para suportar o WS-Addressing:

- O solicitante do serviço da web e os aplicativos do provedor podem interagir com outros serviços que estão usando o WS-Addressing sem precisar reimplementá-los. Um novo manipulador de mensagens, o manipulador de mensagem de endereçamento DFHWSADH, no pipeline, roteia as mensagens que contêm informações do WS-Addressing para o serviço da web especificado.
- É possível gravar um aplicativo que usa os comandos da API do WS-Addressing para criar uma referência de terminal e para criar, atualizar, excluir e consultar um contexto de endereçamento.

- É possível rotear as mensagens de resposta para os terminais diferentes do terminal do solicitante, por exemplo, é possível rotear as mensagens de falha para um manipulador de falha de dedicado.
- É possível transmitir os parâmetros de referência para os aplicativos como parte dos MAPs no cabeçalho SOAP.

Suporte para Especificações e Interoperabilidade do WS-Addressing

Por padrão, o CICS suporta as especificações de recomendação:

- W3C WS-Addressing 1.0 - Núcleo
- W3C WS-Addressing 1.0 - Ligação SOAP
- W3C WS-Addressing 1.0 - Metadados

Essas especificações são identificadas pelo espaço de nomes <http://www.w3.org/2005/08/addressing>. A menos que seja declarado de outra forma, as semânticas do WS-Addressing que estão descritas nesta documentação fazem referência às especificações de recomendação.

Para a interoperabilidade, o CICS também suporta a especificação de envio:

- W3C WS-Addressing - Envio

Esta especificação é identificada pelo espaço de nomes <http://schemas.xmlsoap.org/ws/2004/08/addressing>. Use a especificação de envio somente se você precisar interoperar com um provedor de cliente ou de serviço da web que implementa a especificação de envio.

Visão Geral do Web Services Addressing

Web Services Addressing (WS-Addressing) fornece uma estrutura padrão para especificar os terminais de uma mensagem SOAP. Essa estrutura é neutra de transporte e melhora a interoperabilidade dos serviços da web que utilizam diferentes mecanismos de transporte. A especificação WS-Addressing introduz propriedades de endereçamento de mensagem e referências de terminal.

Web Services Addressing (WS-Addressing) é uma especificação do Worldwide Web Consortium (W3C) que melhora a interoperabilidade entre serviços da web definindo uma maneira padrão de endereçar serviços da web e fornecer informações de endereçamento em mensagens SOAP. As mensagens de SOAP podem ser enviadas por meio de uma variedade de mecanismos de transporte, incluindo HTTP e WebSphere MQ, cada um dos quais armazena informações de destino para a mensagem de uma maneira diferente.

Serviços da web do CICS existentes que são implementados em um pipeline configurado para utilizar WS-Addressing podem utilizar as configurações do WS-Addressing padrão sem requerer nenhuma alteração. Para tirar pleno proveito dos recursos do WS-Addressing, utilize os comandos de API do WS-Addressing. A implementação do WS-Addressing suporta uma falha de SOAP para cada operação WSDL.

Propriedades de Endereçamento de Mensagens

As propriedades de endereçamento de mensagens (MAPs) são um conjunto de propriedades de WS-Addressing bem-definidas que podem ser representadas como elementos em cabeçalhos SOAP. As MAPs fornecem uma maneira padrão de transmitir informações, tal como o terminal para o qual respostas de mensagem devem ser direcionadas ou informações sobre o relacionamento que a mensagem

possui com outras mensagens. As MAPs que são definidas pela especificação WS-Addressing são resumidas na tabela a seguir.

Tabela 11. Propriedades de Endereçamento de Mensagens Definidas pela Especificação WS-Addressing

Nome da MAP do WS-Addressing Abstrata	Nome de MAP de WS-Addressing SOAP	Tipo de conteúdo MAP	Multiplicidade	Descrição
[action]	<wsa:Action>	xs:anyURI	1..1	A URI absoluta que identifica exclusivamente a semântica da mensagem. Esse valor é requerido.
[destino]	<wsa:To>	xs:anyURI na mensagem SOAP EndpointReference no contexto de endereçamento	0..1	O URI absoluto que especifica o endereço do destinatário desejado da mensagem. Se este valor não for especificado, ele será padronizado com o URI anônimo definido na especificação: http://www.w3.org/2005/08/addressing/anonymous . No contexto de endereçamento, a MAP <wsa:To> é representada como um EPR. Quando <wsa:To> é enviado como parte de uma mensagem SOAP, ele é dividido em seu endereço e seus parâmetros de referência, conforme definido pelo esquema.
[reference parameters] *	[reference parameters]*	xs:any	0..ilimitado	Os parâmetros que correspondem às propriedades <wsa:ReferenceParameters> da referência de terminal para a qual a mensagem é endereçada. Este valor é opcional.
[source endpoint]	<wsa:From>	EndpointReference	0..1	Uma referência ao terminal a partir do qual a mensagem se originou. Este valor é opcional.
[reply endpoint]	<wsa:ReplyTo>	EndpointReference	0..1	Uma referência de terminal para o receptor desejado de respostas para esta mensagem. Este valor é opcional. Se esse valor não for especificado, ele será padronizado como http://www.w3.org/2005/08/addressing/anonymous .
[fault endpoint]	<wsa:FaultTo>	EndpointReference	0..1	Uma referência de terminal para o receptor desejado de falhas relacionadas a esta mensagem. Este valor é opcional e padronizado com o valor da MAP <wsa:ReplyTo>.
[relationship] *	<wsa:RelatesTo>	xs:anyURI mais atributo opcional de tipo xs:anyURI	0..ilimitado	Um par de valores que indicam como esta mensagem está relacionada a outra mensagem. O conteúdo deste elemento transmite o <wsa:MessageID> da mensagem relacionada. Um atributo opcional transmite o tipo de relacionamento. Este valor é opcional. Se esse valor não for especificado, ele será padronizado como http://www.w3.org/2005/08/addressing/reply .
[message id]	<wsa:MessageID>	xs:anyURI		Um URI absoluto que identifica exclusivamente a mensagem. Este valor é opcional; se não for fornecido, o CICS gera um valor para solicitações e respostas de saída.

O exemplo a seguir de uma mensagem SOAP contém MAPs do WS-Addressing:

```

<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsa="http://www.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'>ABCDEFG</example:DiscountId>
    ...
  </S:Header>
  <S:Body>
    ...
  </S:Body>
</S:Envelope>

```

Referências de Terminal

Uma referência de terminal é um tipo específico de MAP, que fornece um mecanismo padrão para encapsular informações sobre terminais específicos. As referências de terminal podem ser enviadas para outras partes e utilizadas para direcionar o terminal de serviço da web que elas representam. A tabela a seguir resume o modelo de informações para referências de terminal.

Tabela 12. Modelo de Informações para Referências de Terminal

Nome da Propriedade Abstrata	Tipo de Propriedade	Multiplicidade	Descrição
[endereço]	xs:anyURI	1..1	O URI absoluto que especifica o endereço do terminal.
[reference parameters] *	xs:any	0..ilimitado	Itens de informações de elemento qualificado do namespace que são necessários para criar interface com o terminal.
[metadados]	xs:any	0..ilimitado	Descrição do comportamento, políticas e recursos do terminal.

O fragmento XML a seguir ilustra uma referência de terminal. O elemento <wsa:EndpointReference> faz referência ao terminal no URI http://example.ibm.com/enquiry e contém metadados que especificam a interface à qual a referência de terminal se refere e alguns parâmetros de referência específicos do aplicativo.

```

<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:wSDLi="http://www.w3.org/ns/wsdli-instance"
    wsdli:wSDLLocation="http://example.ibm.com/wsdli/wsdli-location.wsdli">
    <wsam:InterfaceName>example:reservationInterface</wsam:InterfaceName>
  </wsa:Metadata>
  <wsa:ReferenceParameters>

```

```

        <example:AccountCode>123456789</example:AccountCode>
        <example:DiscountId>ABCDEFG</example:DiscountId>
    <wsa:ReferenceParameters>
</wsa:EndpointReference>

```

MAPs WS-Addressing de tipo `wsa:EndpointReferenceType` são: `<wsa:From>`, `<wsa:ReplyTo>` e `<wsa:FaultTo>`. No entanto, a MAP `<wsa:To>` é definida no padrão de WS-Addressing 1.0 como tendo um tipo de `xs:anyURI`. Para simplicidade, o CICS trata MAPs `<wsa:To>` no contexto de endereçamento como EPRs. Quando uma MAP `<wsa:To>` é enviada como parte de uma mensagem SOAP, o CICS a divide em seus parâmetros de endereço e de referência, conforme exigido pelo padrão.

Namespaces Padrão

O prefixo a seguir e os namespaces correspondentes são referidos em toda a documentação do WS-Addressing:

Tabela 13. Prefixo e Namespace Correspondente

Prefixo Padrão	Namespace
xs	http://www.w3.org/2001/XMLSchema
wsa	http://www.w3.org/2005/08/addressing (esquema de Recomendação) http://schemas.xmlsoap.org/ws/2004/08/addressing (esquema de Envio)
wsam	http://www.w3.org/2007/05/addressing/metadata

Configurando um Pipeline do Solicitante para Web Services Addressing

Para configurar um pipeline do solicitante para suportar Web Services Addressing (WS-Addressing), você deve incluir um manipulador de endereçamento em seu arquivo de configuração de pipeline.

Antes de Iniciar

Você deve identificar ou criar o arquivo de configuração de pipeline para incluir as informações de configuração para WS-Addressing. Você também deve decidir qual das especificações WS-Addressing usar. Use a especificação *W3C WS-Addressing 1.0 Core* onde possível.

Sobre Esta Tarefa

É possível incluir suporte para WS-Addressing de uma de duas maneiras:

- Se o pipeline SOAP usar Java, o processamento de SOAP será manipulado por Axis2 e será possível usar o suporte fornecido por esta tecnologia para manipular solicitações que usam WS-Addressing. Toda manipulação do cabeçalho é manipulada por Axis2 e é importante que você não inclua o programa de processamento de cabeçalho DFHWSADH no pipeline. É possível usar seus próprios programas de processamento de cabeçalho. Para melhor desempenho, grave manipuladores Axis2 em Java se desejar processar cabeçalhos SOAP.
- Se o pipeline SOAP não usar Java, você deverá incluir o programa de processamento de cabeçalho fornecido pelo CICS, DFHWSADH, para manipular as solicitações.

Procedimento

- Se o pipeline SOAP usa um elemento `<cics_soap_1.1_handler_java>` ou `<cics_soap_1.2_handler_java>`, inclua um elemento `<addressing>` no arquivo de configuração de pipeline. Inclua um elemento `<namespace>` que contém a especificação que deseja usar na mensagem de solicitação, que pode ser diferente para a mensagem de resposta; por exemplo, é possível enviar sempre uma solicitação que esteja em conformidade com a especificação de núcleo W3C, mesmo se a mensagem de resposta usar a especificação de envio. Axis2 suporta ambas as especificações WS-Addressing em mensagens de entrada.

O exemplo a seguir mostra como você pode configurar o pipeline do 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>
```

O elemento `<jvmserver>` contém o nome do recurso JVMSEV1 que suporta Axis2.

- Se o pipeline SOAP não usar Java, inclua um programa do cabeçalho de endereçamento CICS no `<cics_soap_1.1_handler>` ou `<cics_soap_1.2_handler>` para o arquivo de configuração de pipeline. O exemplo a seguir mostra como você pode configurar o pipeline do 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 os elementos `<program_name>`, `<localname>` e `<mandatory>` exatamente conforme mostrado. Configure `<namespace>` como `http://www.w3.org/2005/08/addressing` para usar a especificação *W3C WS-Addressing 1.0 Core* ou `http://schemas.xmlsoap.org/ws/2004/08/addressing` para usar a especificação *W3C WS-Addressing Submission*.

A ordem dos programas de processamento de cabeçalho não é garantida. Se definir outros programas de processamento de cabeçalho, inclua-os em um elemento manipulador CICS SOAP subsequente em seu elemento `<service_handler_list>`. O manipulador de cabeçalho DFHWSADH deve estar no primeiro elemento do manipulador SOAP.

Resultados

Seu pipeline do solicitante agora é configurado para suportar WS-Addressing.

O que Fazer Depois

Crie um recurso PIPELINE que aponta para o arquivo de configuração. Se estiver usando um pipeline SOAP baseado em Java, assegure que um recurso JVMLSERVER esteja ativado para manipular o processamento do Axis2.

Configurando um Pipeline do Provedor para Web Services Addressing

Para configurar um pipeline do provedor para suportar Web Services Addressing (WS-Addressing), você deve incluir um manipulador de endereçamento em seu arquivo de configuração de pipeline.

Antes de Iniciar

Você deve identificar ou criar o arquivo de configuração de pipeline para incluir as informações de configuração para WS-Addressing. Você também deve decidir qual das especificações WS-Addressing usar. Use a especificação *W3C WS-Addressing 1.0 Core* onde possível.

Sobre Esta Tarefa

É possível incluir suporte para WS-Addressing de uma de duas maneiras:

- Se o pipeline SOAP usar Java, o processamento de SOAP será manipulado por Axis2 e será possível usar o suporte fornecido por esta tecnologia para manipular solicitações que usam WS-Addressing. Toda manipulação do cabeçalho é manipulada por Axis2 e é importante que você não inclua o programa de processamento de cabeçalho DFHWSADH no pipeline. É possível usar seus próprios programas de processamento de cabeçalho. Para melhor desempenho, grave manipuladores Axis2 em Java se desejar processar cabeçalhos SOAP.
- Se o pipeline SOAP não usar Java, você deverá incluir o programa de processamento de cabeçalho fornecido pelo CICS, DFHWSADH, para manipular as solicitações.

Procedimento

- Se o pipeline SOAP usa um elemento `<cics_soap_1.1_handler_java>` ou `<cics_soap_1.2_handler_java>`, inclua um elemento `<addressing>` no arquivo de configuração de pipeline. É possível, opcionalmente, incluir um ou mais elementos `<namespace>`. Este elemento contém a especificação que você deseja usar na mensagem de saída, que pode ser diferente para a mensagem de entrada; por exemplo, é possível sempre enviar uma resposta de saída que está em conformidade com a especificação de núcleo de W3C, mesmo se a mensagem de entrada usar a especificação de envio. Se você excluir este elemento, Axis2 usará a mesma especificação na mensagem de saída que a mensagem de entrada. Axis2 suporta ambas as especificações WS-Addressing em mensagens de entrada.

O exemplo a seguir mostra como você pode configurar o pipeline do provedor:

```
<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>

```

O elemento <jvmserver> contém o nome do recurso JVMSERVER que suporta Axis2.

- Se o pipeline SOAP não usar Java, inclua o programa do cabeçalho de endereçamento do CICS, DFHWSADH, no manipulador SOAP no arquivo de configuração de pipeline. O exemplo a seguir mostra como você pode configurar o pipeline do provedor:

```

<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 os elementos <program_name>, <localname> e <mandatory> exatamente conforme mostrado. Configure <namespace> como <http://www.w3.org/2005/08/addressing> para usar a especificação *W3C WS-Addressing 1.0 Core* ou <http://schemas.xmlsoap.org/ws/2004/08/addressing> para usar a especificação *W3C WS-Addressing Submission*.

A ordem dos programas de processamento de cabeçalho não é garantida. Se você definir outros programas de processamento de cabeçalho, inclua-os em um outro elemento do manipulador SOAP do CICS em um elemento <service_handler_list>. O manipulador de cabeçalho DFHWSADH deve estar no último elemento manipulador SOAP.

Resultados

Seu pipeline do provedor agora está configurado para suportar WS-Addressing.

O que Fazer Depois

Crie um recurso PIPELINE que aponta para o arquivo de configuração. Se estiver usando um pipeline SOAP baseado em Java, assegure que um recurso JVMSERVER esteja ativado para manipular o processamento do Axis2.

Criando um Serviço da Web que Usa WS-Addressing

Para criar um serviço da web a partir de um documento WSDL que usa Web Services Addressing (WS-Addressing), utilize os parâmetros no assistente de serviços da web para manipular a conversão de XML para estruturas de linguagem.

Sobre Esta Tarefa

É possível utilizar a tarefa do assistente de serviços da web, DFHWS2LS, para controlar como uma referência de terminal (EPR) é manipulada no documento WSDL e determinar se o CICS constrói ações de entrada, de saída e de falha padrão.

Procedimento

1. Configure o parâmetro **MINIMUM-RUNTIME** no assistente de serviços da web, DFHWS2LS, como 3.0 ou superior. Um nível de tempo de execução de pelo menos 3,0 garante que qualquer ligação de serviço da web gerada suporte totalmente WS-Addressing e possa interoperar com outras plataformas de serviços da web.
2. Configure o parâmetro **MAPPING-LEVEL** no assistente de serviços da web, DFHWS2LS, como 3.0 ou superior.
3. Configure o parâmetro **WSADDR-EPR-ANY** como TRUE se você desejar usar os elementos de tipo `wsa:EndpointReferenceType` nas mensagens de solicitação ou resposta. As referências de terminal podem ser incluídas em dados do aplicativo e você tem a opção de utilizar a EPR em comandos de API como **WSACONTEXT BUILD**. A configuração do parâmetro **WSADDR-EPR-ANY** como TRUE indica que o CICS não deve transformar a EPR em uma estrutura de linguagem no tempo de execução; em vez disso, o CICS trata os dados de EPR como um elemento `<xsd:any>` e o armazena em um contêiner nomeado.

Este fragmento de WSDL de exemplo mostra um MAP `<wsa:To>` sendo transmitido como um elemento do 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"/> ❶
        </xs:sequence>
      </xs:complexType>
      ...
    </xs:schema>
  </types>
  ...
  <message name="msgResponse">
    <part element="s0:exampleResponse" name="response"/>
  </message>
  ...
</definitions>
```

Quando o elemento, `<xs:element name="myEpr" type="wsa:EndpointReferenceType"/>` ❶, é processado por DFHWS2LS com o parâmetro **WSADDR-EPR-ANY** configurado como TRUE, os dados do elemento `myEpr` são armazenados em um contêiner nomeado como um elemento `<xsd:any>` e um ponteiro para o contêiner incluído na estrutura de linguagem gerada.

Por exemplo, a estrutura de linguagem COBOL gerada por DFHWS2LS para o elemento `myEpr` é mostrada aqui:

```
09 myEpr.
   12 myEpr-xml-cont          PIC X(16).
   12 myEpr-xmlns-cont        PIC X(16).
```

O contêiner myEpr-xml-cont armazena o nome do contêiner que contém os dados myEpr. myEpr-xmlns-cont é um contêiner opcional que é preenchido com quaisquer declarações de namespace XML que estão no escopo.

4. Salve e envie a tarefa DFHWS2LS.

Resultados

O CICS cria uma ligação de serviço da web para manipular a transformação de dados e as estruturas de linguagem que podem ser usadas para criar o aplicativo solicitante ou provedor de serviço.

O que Fazer Depois

Para ativar o serviço da web, execute uma varredura de pipeline para criar os recursos do CICS necessários.

Referências de Terminal Padrão:

A maioria dos documentos WSDL contém o endereço no qual o serviço da web é hospedado. No WS-Addressing, o documento WSDL também pode conter uma referência de terminal (EPR) para o serviço da web. Esta EPR pode conter metadados adicionais para facilitar a comunicação entre os aplicativos do provedor e do solicitante.

Se usar DFHWS2LS para processar o WSDL, a EPR será salva na ligação de serviço da web e será usada pelo CICS para enviar mensagens de solicitação e resposta. Quaisquer parâmetros de referência, <wsa:ReferenceParameters>, que são definidos na EPR são incluídos na mensagem SOAP. Esta EPR é conhecida como a EPR padrão, porque ela pode ser substituída pelo aplicativo. Se o aplicativo não fornecer uma EPR explícita, a EPR padrão do WSDL será usada.

O fragmento WSDL 1.1 a seguir inclui uma EPR padrão: <soap:address location="http://example.ibm.com:12345/exampleTest" />. O elemento <port> inclui um elemento filho, <wsa:EndpointReference>, o endereço especificado pelo elemento filho, **2**, deve corresponder ao endereço especificado pelo elemento-pai, **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:w3="http://www.w3.org/2006/01/wsdl-instance"
      w3:w3Location="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>ABCDEFGH</example:DiscountId>
        </wsa:ReferenceParameters>
      </wsa:EndpointReference>
    </port>
  </service>
```

Ações Explícitas:

Os documentos WSDL podem definir explicitamente os valores das propriedades <wsa:Action>. Se o documento WSDL não contém propriedades <wsa:Action> definidas explicitamente, o CICS constrói ações padrão quando o WSDL é processado pelo DFHWS2LS.

WSDL 1.1

O fragmento WSDL 1.1 a seguir representa um sistema de registro que contém propriedades <wsa:Action> definidas explicitamente:

```
<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>
```

Neste exemplo, a ação de entrada da operação makeBooking é definida explicitamente como http://example.ibm.com/namespace/makeBooking e a ação de saída é definida explicitamente como http://example.ibm.com/namespace/bookingResponse.

WSDL 2.0

O fragmento WSDL 2.0 a seguir representa um sistema de registro que contém propriedades <wsa:Action> definidas explicitamente:

```
<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"/>
      <output element="tns:makeBookingResponse" messageLabel="Out"
        wsa:Action="http://example.ibm.com/namespace/makeBookingResponse"/>
    </operation>
  </interface>
  ...
</description>
```

Neste exemplo, a ação de entrada da operação makeBooking é definida explicitamente como http://example.ibm.com/namespace/makeBooking e a ação de saída é definida como http://example.ibm.com/namespace/makeBookingResponse.

Para obter mais informações, consulte a especificação W3C WS-Addressing 1.0 Metadata.

Ações Padrão para WSDL 1.1:

Se um documento WSDL 1.1 não contém propriedades <wsa:Action> especificadas explicitamente, o CICS constrói a entrada, a saída e as ações de falha padrão quando o WSDL é processado pelo DFHWS2LS.

Ações de Entrada e Saída Padrão para WSDL 1.1

O padrão a seguir é usado pelo CICS nos documentos WSDL 1.1 que seguem o esquema de recomendação ou o esquema de envio para construir uma ação de entrada ou saída padrão:

[namespace de destino]/[nome do tipo de porta]/[nome de entrada|saída]

Ações de Falha Padrão para WSDL 1.1

Se estiver seguindo o esquema de recomendação, a maneira como o CICS constrói a ação de falha padrão difere do comportamento descrito no esquema. O padrão a seguir é usado pelo CICS, em documentos WSDL 1.1 que seguem o esquema de recomendação, para construir uma mensagem de falha padrão. Observe que o nome da falha é omitido.

[namespace de destino]/[nome do tipo de porta]/[nome da operação]/Falha/

Se estiver seguindo o esquema de envio, a maneira como o CICS constrói a ação de falha padrão segue o comportamento descrito no esquema. O padrão a seguir é usado pelo CICS, em documentos WSDL 1.1 que seguem o esquema de envio, para construir uma mensagem de falha padrão:

[namespace de destino]/[nome do tipo de porta]/[nome da operação]/Falha/[nome da falha]

Exemplo das Ações Padrão Geradas pelo CICS para um Documento WSDL 1.1

Este exemplo de um sistema de registro ilustra como o CICS constrói ações padrão a partir de um 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>
</portType>
</interface>
...
</definitions>
```

O fragmento de WSDL possui as propriedades de endereçamento a seguir:

O nome da propriedade	Mínimo
[targetNamespace]	http://example.ibm.com/namespace
[portType name]	bookingInterface
[operation name]	makeBooking
[input name]	MakeBooking
[output name]	BookingResponse
[fault name]	InvalidBooking

As ações a seguir são criadas a partir destes valores:

Ação	Mínimo
Ação de Entrada	<p><code>http://example.ibm.com/namespace/bookingInterface/MakeBooking</code></p> <p>Se o [nome de entrada] não for especificado, o valor do [nome da operação] com "Request" anexado será usado em seu lugar. Por exemplo, neste caso a Ação de Entrada é <code>http://example.ibm.com/namespace/bookingInterface/makeBookingRequest</code>.</p> <p><code>http://example.ibm.com/namespace/bookingInterface/BookingResponse</code></p>
Ação de Saída	<p>Se o [nome de saída] não for especificado, o valor do [nome da operação] com "Response" anexado é usado em seu lugar. Por exemplo, neste caso a Ação de Saída é <code>http://example.ibm.com/namespace/bookingInterface/makeBookingResponse</code></p>
Ação de Falha (Esquema de Recomendação)	<p><code>http://example.ibm.com/namespace/bookingInterface/MakeBooking/Fault/</code></p> <p>Observe que o [nome da falha] é omitido.</p>
Ação de Falha (Esquema do Envio)	<p><code>http://example.ibm.com/namespace/bookingInterface/MakeBooking/Fault/InvalidBooking</code></p>

Para obter mais informações, consulte a especificação W3C WS-Addressing 1.0 Metadata.

Ações Padrão para WSDL 2.0:

Se um documento WSDL 2.0 não contiver propriedades `<wsa:Action>` especificadas explicitamente, o CICS construirá ações de entrada, saída e falha padrão quando o WSDL for processado por DFHWS2LS.

Ações de Entrada e Saída Padrão para WSDL 2.0

O padrão a seguir é usado pelo CICS, em documentos WSDL 2.0 que seguem o esquema de recomendação, para construir ações padrão para entradas e saídas:

`[namespace de destino]/[nome da interface]/[nome da operação][token de direção]`

Ações de Falha Padrão para WSDL 2.0

Se estiver seguindo o esquema de recomendação, a maneira que o CICS constrói a ação padrão para falhas de WS-Addressing difere do comportamento descrito no esquema. Se estiver seguindo o esquema de envio, a maneira como o CICS constrói a ação padrão para falhas de WS-Addressing segue o comportamento descrito no esquema.

O padrão a seguir é usado pelo CICS, em documentos WSDL 2.0 que seguem o esquema de recomendação, para construir uma ação padrão para falhas. Observe que o nome da falha é omitido.

`[namespace de destino]/[nome da interface]/`

O padrão a seguir é usado pelo CICS, em documentos WSDL 2.0 que seguem o esquema de envio, para construir uma ação padrão para falhas:

`[namespace de destino]/[nome da interface]/[nome da falha]`

Exemplo das Ações Padrão Geradas pelo CICS para um Documento WSDL 2.0

Este exemplo mostra como o CICS constrói ações padrão para um documento WSDL 2.0 seguindo o esquema de recomendação:

```

<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>

```

O fragmento de WSDL possui as propriedades de endereçamento a seguir:

Nome da propriedade	Mínimo
[targetNamespace]	http://example.ibm.com/namespace
[interface name]	bookingInterface
[operation name]	makeBooking
[direction token]	Solicitação ou Resposta.

As ações de entrada e saída a seguir são criadas a partir destes valores:

Ação	Mínimo
Ação de Entrada	http://example.ibm.com/namespace/bookingInterface/makeBookingRequest
Ação de Saída	http://example.ibm.com/namespace/bookingInterface/makeBookingResponse

Para obter mais informações, consulte a especificação W3C WS-Addressing 1.0 Metadata.

Trocas de Mensagens

O Web Services Addressing (WS-Addressing) suporta estas trocas de mensagens: unidirecional, solicitação-resposta bidirecional, solicitação-resposta síncronas e solicitação-resposta assíncronas.

As trocas de mensagens do Web Services Addressing envolvem propriedades de endereçamento de mensagens (MAPs) e referências de terminal (EPRs).

Em tempo de execução, o CICS assegura que o cabeçalho SOAP da mensagem de solicitação contenha as informações de mensagem do WS-Addressing relevantes, o aplicativo do solicitante não precisa configurar os cabeçalhos WS-Addressing e podem não estar cientes que o WS-Addressing está sendo usado.

Unidirecional

Esta mensagem unidirecional direta é definida como uma operação somente de entrada. O Web Services Description Language (WSDL) para esta operação tem o seguinte formato:

```

<operation name="myOperation">
  <input message="tns:myInputMessage"/>
</operation>

```

Se você estiver utilizando WS-Addressing, o CICS incluirá as MAPs <wsa:Action> e a MAP <wsa:MessageID> para o cabeçalho da mensagem SOAP da mensagem de solicitação de WS-Addressing em tempo de execução para assegurar a conformidade com a especificação WS-Addressing.

A MAP <wsa:MessageID> é um ID exclusivo, se não especificado, o CICS gerará este ID automaticamente.

As MAPs <wsa:Action> são derivadas do WSDL e armazenadas no arquivo WSBind.

É possível substituir os valores destas MAPs usando os comandos da API do WS-Addressing do CICS.

Solicitação-Resposta Bidirecional

Essa troca bidirecional envolve uma mensagem de solicitação e uma mensagem de resposta. A parte da resposta da operação pode ser definida como uma mensagem de saída, uma mensagem de falha ou ambas. A definição de WSDL para a operação de solicitação-resposta tem o seguinte formato:

```
<operation name="myOperation">
  <input message="tns:myInputMessage"/>
  <output message="tns:myOutputMessage"/>
  <fault="tns:myFaultMessage"/>
</operation>
```

As respostas a, ou falhas geradas a partir de, solicitações que são direcionados em terminais são destinadas na MAP <wsa:ReplyTo> ou na MAP <wsa:FaultTo>, dependendo de se o tipo de resposta é normal ou uma falha.

Especifique uma MAP <wsa:ReplyTo> ou <wsa:FaultTo> na mensagem de solicitação para indicar onde a resposta deve ser enviada.

Se você estiver utilizando as especificações de recomendação e não especificar um valor para a MAP <wsa:ReplyTo>, a MAP <wsa:ReplyTo> será padronizada com uma referência de terminal que contém o URI anônimo (<http://www.w3.org/2005/08/addressing/anonymous>), que faz com que o CICS envie a resposta de volta para o solicitante.

Se estiver usando as especificações de recomendação e não especificar um valor para a MAP <wsa:FaultTo>, a MAP <wsa:FaultTo> padronizará o valor da MAP <wsa:ReplyTo>.

Se o solicitante construir MAPs que estão incorretas e que causam falhas de validação, o CICS enviará a mensagem de falha de volta ao solicitante em vez de para o endereço especificado pela MAP <wsa:FaultTo>.

Solicitação-Resposta Síncronas

Por padrão, a parte da resposta de uma mensagem bidirecional será retornada de acordo com o protocolo subjacente em uso. No caso de uma solicitação de HTTP, a resposta é retornada sincronicamente na resposta HTTP.

Solicitação-Resposta Assíncronas

Uma resposta assíncrona é destinada a outro serviço da web e não chega de volta no aplicativo solicitante original. No caso de uma solicitação de HTTP, a conexão com o cliente solicitante é fechada com uma resposta HTTP 202. Se o provedor de serviços da web estiver sendo executado em um sistema CICS, o aplicativo solicitante receberá uma mensagem de resposta vazia. Se o provedor de serviço da

web estiver em execução em um sistema do WebSphere MQ, o aplicativo do solicitante não receberá nenhuma resposta.

Para alterar o destino da parte da resposta de uma mensagem bidirecional, você deve especificar os endereços apropriados na MAP <wsa:ReplyTo> ou as MAPs <wsa:ReplyTo> e <wsa:FaultTo>.

Para obter uma lista completa das MAPs que são obrigatórias no WSDL 1.1 e no WSDL 2.0, consulte “Propriedades de Endereçamento de Mensagens Obrigatórias para WS-Addressing”.

Propriedades de Endereçamento de Mensagens Obrigatórias para WS-Addressing

A especificação de metadados WS-Addressing 1.0 indica quais propriedades de endereçamento de mensagens (MAPs) devem ser incluídas nos documentos WSDL 1.1 e WSDL 2.0. A implementação do CICS de WS-Addressing o ajuda a estar em conformidade com as especificações WS-Addressing fornecendo automaticamente valores para essas MAPs obrigatórias.

É possível especificar seus próprios valores para MAPs no WSDL fornecido e você pode atualizar estes valores no contexto de endereçamento usando os comandos da API de WS-Addressing do CICS. Se você não fornecer valores para as MAPs obrigatórias, o CICS gerará valores para você.

A tabela a seguir lista quais MAPs são obrigatórias para os diferentes padrões de troca de mensagens (MEPs) suportados com WSDL 1.1 e WSDL 2.0:

Tabela 14. Propriedades de Endereçamento de Mensagens Obrigatórias para WS-Addressing.

Nome de MAP do WS-Addressing	Descrição	Obrigatória no WSDL 1.1	Obrigatória no WSDL 2.0
<wsa:To>	O endereço do destinatário desejado da mensagem.	Não	Não
<wsa:Action>	A ação de WS-Addressing: entrada, saída ou falha.	Obrigatório para os MEPs a seguir: Unidirecional Bidirecional (Solicitação) Bidirecional (Resposta)	Obrigatório para os MEPs a seguir: In-only Robust In-only (Entrada) Robust In-only (Falha) In-out (Entrada) In-out (Saída) In-optional-out (Entrada) In-optional-out (Saída)
<wsa:From>	O terminal a partir do qual a mensagem se originou.	Não	Não
Isto	value	não igual	Requerido
<wsa:ReplyTo>	O terminal do destinatário desejado para respostas à mensagem.	Não	Não
<wsa:FaultTo>	O terminal do destinatário desejado para falhas relacionadas à mensagem.	Não	Não

Tabela 14. Propriedades de Endereçamento de Mensagens Obrigatórias para WS-Addressing. (continuação)

Nome de MAP do WS-Addressing	Descrição	Obrigatória no WSDL 1.1	Obrigatória no WSDL 2.0
<wsa:MessageID>	Um identificador de mensagem exclusivo.	Obrigatório para os MEPs a seguir: Bidirecional (Solicitação)	Obrigatório para os MEPs a seguir: Robust In-only (Entrada) In-out (Entrada) In-optional-out (Entrada)
<wsa:RelatesTo>	Um par de valores que indicam como esta mensagem se relaciona a uma outra mensagem. Este elemento inclui o <wsa:MessageID> da mensagem relacionada e um atributo opcional transmite o tipo de relacionamento.	Obrigatório para os MEPs a seguir: Bidirecional (Resposta)	Obrigatório para os MEPs a seguir: Robust In-only (Falha) In-out (Saída) In-optional-out (Saída)

Para obter mais informações, consulte a especificação W3C WS-Addressing 1.0 Metadata: <http://www.w3.org/TR/ws-addr-metadata>.

Notas:

- Se um valor não for configurado para o elemento de endereço da MAP <wsa:ReplyTo>, o endereço será configurado com o URI anônimo: <http://www.w3.org/2005/08/addressing/anonymous>. O URI anônimo indica que as respostas são enviadas de volta ao solicitante.
- Se um valor não for especificado para o elemento de endereço da MAP <wsa:FaultTo>, o CICS configurará este endereço com o mesmo valor que o elemento de endereço da MAP <wsa:ReplyTo>.
Observe que, se o solicitante construir MAPs que estão incorretas e que causam falhas de validação, o CICS enviará a mensagem de falha de volta ao solicitante em vez de ao endereço especificado pela MAP <wsa:FaultTo>.
- Se o valor da MAP <wsa:To> não for especificado, o CICS configurará o endereço com o URI anônimo: <http://www.w3.org/2005/08/addressing/anonymous>. O URI anônimo indica que a solicitação deve ser enviada ao endereço especificado no contêiner DFHWS-URI; para obter mais informações, consulte “Contêiner DFHWS-URI” na página 164.
- É possível definir as MAPs <wsa:Action> explicitamente em seu documento WSDL ou você pode deixar que o CICS as gere automaticamente.
- O CICS configura automaticamente um valor exclusivo para a MAP <wsa:MessageID> no tempo de execução para mensagens de solicitação que esperam uma resposta e para mensagens de resposta.
- A MAP <wsa:RelatesTo> é obrigatória para mensagens de resposta. O tipo de relacionamento da mensagem é opcional e padronizado como <http://www.w3.org/2005/08/addressing/reply>.

Segurança de Web Services Addressing

As comunicações que viajam em uma rede pública usando o Web Services Addressing (WS-Addressing) devem ser protegidas adequadamente e um nível suficiente de confiança deve ser estabelecido entre as partes da comunicação. É recomendado que você use a segurança no nível de transporte, tal como SSL ou HTTPS, para proteger suas comunicações.

A segurança no nível do transporte, tal como SSL ou HTTPS, é a maneira mais direta de assegurar que suas comunicações do WS-Addressing sejam seguras. Se a segurança no nível do transporte não estiver disponível, será possível proteger suas mensagens assinando as propriedades de endereçamento de mensagens do WS-Addressing e criptografando as referências de terminal.

O CICS não pode assinar cabeçalhos contendo propriedades de endereçamento de mensagens do WS-Addressing ou criptografar referências de terminal. No entanto, o CICS pode verificar assinaturas em mensagens recebidas e podem decriptografar cabeçalhos que foram criptografados. Se desejar usar assinatura e criptografia para proteger suas comunicações, você deverá usar um gateway de segurança externo, tal como o IBM WebSphere DataPower XML Security Gateway. Para obter mais informações, consulte .

Exemplo de Web Services Addressing

Este exemplo fornece uma visão geral de alto nível do processo que ocorre quando um cliente faz um pedido com uma empresa que utiliza o Web Services Addressing para enviar mensagens.

Uma empresa internacional que vende componentes eletrônicos usa o Web Services Addressing em seus negócios. A infraestrutura desta empresa consiste em um Cliente de Pedido, um grupo de Serviços de Distribuição, um Serviço de Finalização e um Serviço de Configuração.

O uso do WS-Addressing oferece à empresa os benefícios a seguir:

- O WS-Addressing fornece um mecanismo independente de transporte para transferir mensagens, isto encoraja a interoperabilidade entre serviços da web que são executados em diferentes plataformas. Neste exemplo, os serviços de distribuição pertencentes à empresa estão sendo executados em uma variedade de plataformas; WS-Addressing faz a interoperabilidade entre plataformas diferentes diretamente porque os solicitantes e provedores de serviços da web não precisam estar cientes da plataforma na qual o serviço com o qual eles estão trocando mensagens está em execução.
- O WS-Addressing pode ser utilizado para alterar o destino da mensagem de resposta atualizando o EPR na MAP <wsa:ReplyTo>. Neste exemplo, o Serviço de Finalização modifica o destino da mensagem de resposta quando ele seleciona o Serviço de Distribuição para o qual a mensagem será desviada.

A empresa possui vários centros de distribuição em vários países diferentes; cada um dos centros de distribuição é representado neste exemplo por um Serviço de Distribuição e é registrado com o Serviço de Configuração.

O Serviço de Finalização seleciona qual serviço de Distribuição é o mais apropriado para processar o pedido com base em uma variedade de fatores, que podem incluir a disponibilidade de itens solicitados e a distância do Centro de Distribuição do cliente.

Informações de endereçamento são transmitidas para e a partir do Serviço de Configuração. O Serviço de Configuração armazena os endereços dos serviços disponíveis na forma de Referências de Terminal. Novo registro de serviços com o Serviço de Configuração criando um EPR usando o comando **WSAEPR CREATE** e enviando o EPR ao Serviço de Configuração. O Serviço de Configuração requer o EPR como um bloco de XML, portanto, o parâmetro **WSADDR-EPR-ANY** em **DFHWS2LS** deve ser configurado como **TRUE**. A opção **WSADDR-EPR-ANY=TRUE** é

usada para instruir o CICS a tratar o EPR como um elemento <xsd:any>; o CICS deve colocá-lo em um contêiner em vez de transformá-lo em uma estrutura de linguagem no tempo de execução.

A maneira na qual estes serviços interagem é mostrada no diagrama a seguir. O diagrama mostra outros serviços, que foram excluídos da tarefa, que podem ser relevantes em um aplicativo de negócios:

- Um Serviço de Rastreamento, que pode ser atualizado por cada um dos outros serviços com o status do pedido.
- Um serviço de Resolução de Problemas para manipular todas as mensagens de falha que surgem.
- Um serviço de retorno de chamada do Cliente de Pedido para manipular quaisquer mensagens de resposta direcionadas no Cliente de Pedido.

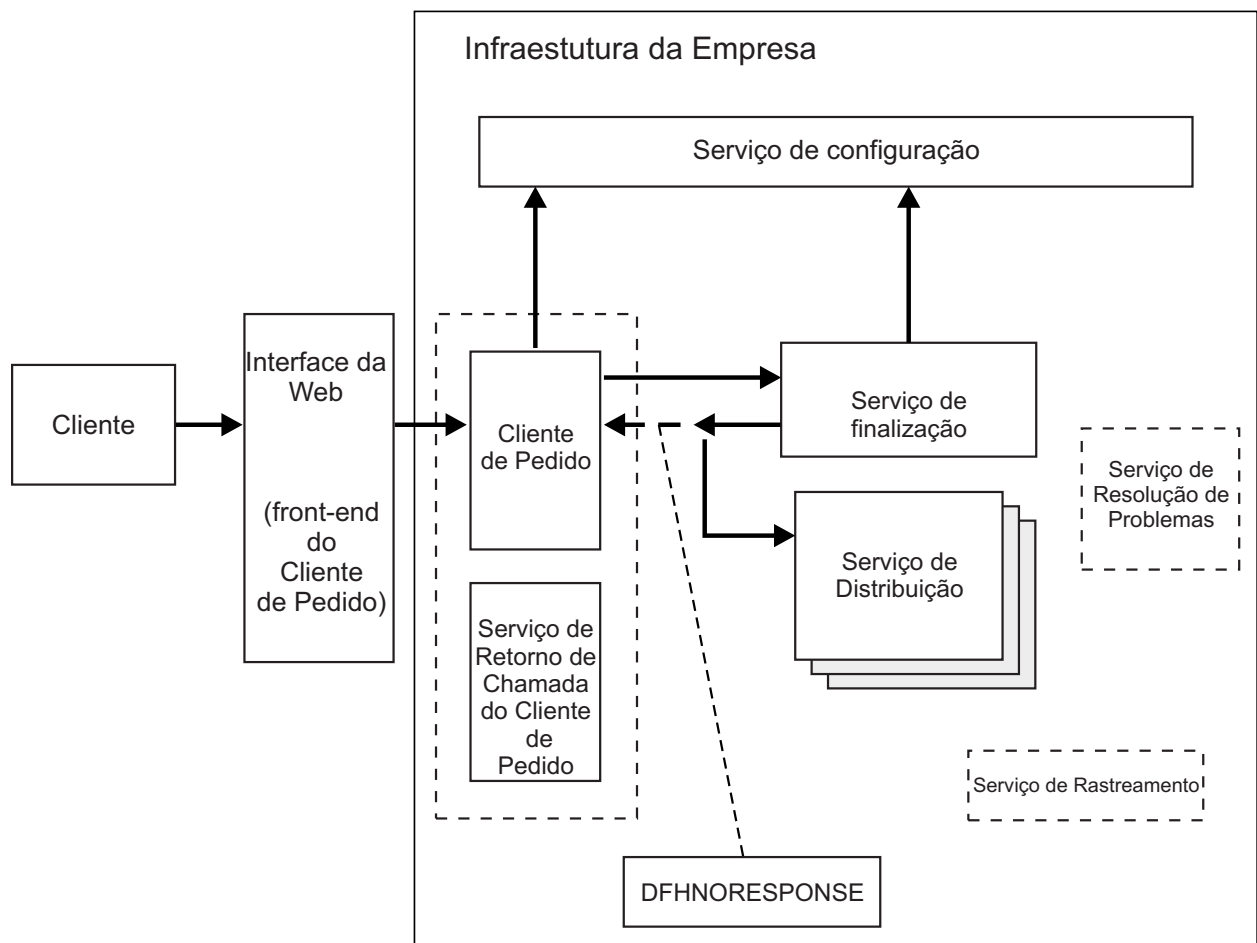


Figura 26. Infraestrutura da Empresa

As etapas a seguir descrevem o processo que ocorre a partir do momento que um cliente faz um pedido até o ponto no qual esse pedido é processado.

1. Um cliente faz um pedido com a empresa.
 - a. O cliente efetua o pedido no website da empresa, que é o front-end para o Cliente de Pedido.

- b. O Cliente de Pedido utiliza os detalhes de contato do cliente como parte do pedido.
 - c. O Cliente de Pedido retorna uma confirmação e uma referência de pedido exclusiva para o cliente por meio da interface da web.
- 2. O Cliente de Pedido envia a solicitação do pedido para o Serviço de Finalização.
 - a. Se o Cliente de Pedido ainda não conhecer o EPR para o Serviço de Finalização, ele o solicitará a partir do Serviço de Configuração. O processo envolvido quando o Cliente de Pedido solicita o EPR do Serviço de Finalização a partir do Serviço de Configuração é detalhado na seção Exemplo de <wsa:To>.
 - b. O Cliente de Pedido emite o comando **INVOKE SERVICE** para o Serviço de Finalização. O WS-Addressing roteia a mensagem para o endereço especificado pelo EPR To no contexto de endereçamento da solicitação.
- 3. O Serviço de Finalização seleciona um Serviço de Distribuição para processar o pedido e redireciona a mensagem de resposta para esse serviço.
 - a. O Serviço de Finalização usa um comando **WSACONTEXT GET** para extrair a referência do pedido e outras propriedades de endereçamento a partir do contexto de endereçamento.
 - b. O Serviço de Finalização seleciona o Serviço de Distribuição mais apropriado no Serviço de Configuração.
 - c. O EPR <wsa:ReplyTo> é incluído no contexto de endereçamento:


```
<wsa:EndpointReference
  xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <wsa:Address>http://www.example.ibm.com/DistributionService</wsa:Address>
</wsa:EndpointReference>
```

O Serviço de Finalização utiliza o comando **WSACONTEXT BUILD** para incluir o EPR ReplyTo do Serviço de Distribuição escolhido no contexto de endereçamento da solicitação.
 - d. O Serviço de Finalização usa o comando **WSACONTEXT BUILD** repetidamente para incluir a referência de pedido e outras informações no contexto de endereçamento da solicitação.
 - e. Um contêiner DFHNORESPONSE é incluído no pipeline do Cliente de Pedido para indicar ao Cliente de Pedido que ele não receberá uma resposta e a mensagem de resposta será redirecionada na forma de uma mensagem de solicitação ao Serviço de Distribuição.
- 4. O Serviço de Distribuição recebe a mensagem de resposta redirecionada e processa o pedido.
 - a. O Serviço de Distribuição usa um comando **WSACONTEXT GET** para extrair a referência do pedido e os detalhes de endereçamento do contexto de endereçamento da solicitação.
 - b. O Serviço de Distribuição processa o pedido.

Exemplo de <wsa:To>

- 1. O Cliente de Pedido solicita o EPR do serviço para o qual ele deseja enviar uma mensagem a partir do Serviço de Configuração. Neste exemplo, o Cliente de Pedido solicita o EPR do Serviço de Finalização.
- 2. O Serviço de Configuração cria e envia uma mensagem de resposta:
 - a. O Serviço de Configuração cria o EPR <wsa:To> solicitado para o Serviço de Finalização usando o comando da API **WSAEPR CREATE: EXEC CICS WSAEPR CREATE**.

- b. O Serviço de Configuração grava a saída do comando **WSAEPR CREATE** em um contêiner: EXEC CICS PUT CONTAINER(work-cont).
- c. O Serviço de Configuração copia o nome do contêiner no elemento myEpr-xml-cont: MOVE work-cont TO myEpr-xml-cont.
- d. O Serviço de Configuração envia uma mensagem de resposta ao Cliente de Pedido, esta mensagem contém o conteúdo do contêiner nomeado pelo contêiner myEpr-xml-cont. Neste exemplo, o conteúdo do contêiner work-cont é enviado ao Cliente de Pedido dentro do 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 mostra a troca de mensagem de solicitação-resposta entre o Cliente de Pedido e o Serviço de Configuração. Esta troca de mensagem envolve dois pipelines de serviços da web típicos.

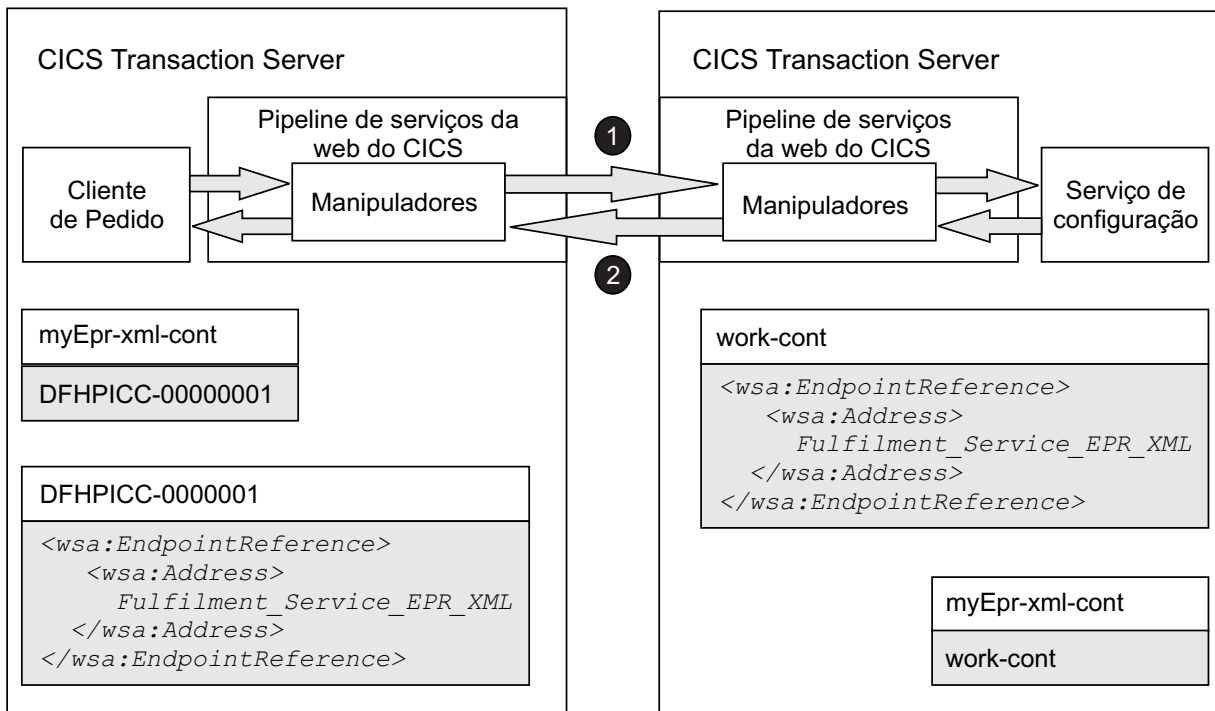


Figura 27. Troca de Mensagem de Solicitação-Resposta Entre o Cliente de Pedido e o Serviço de Configuração

3. O Cliente de Pedido recebe a mensagem de resposta, constrói o EPR <wsa:To> e envia uma solicitação ao Serviço de Finalização:
 - a. O Cliente de Pedido extrai os dados do EPR <wsa:To> da mensagem de resposta.

- b. O CICS preenche um contêiner exclusivo, neste exemplo o contêiner DFHPICC-00000001, com os dados EPR <wsa:To>.
- c. O CICS copia o nome do contêiner, neste exemplo DFHPICC-00000001, no elemento myEpr-xml-cont.
- d. O Cliente de Pedido lê o conteúdo do contêiner especificado pelo elemento myEpr-xml-cont e fornece isso como entrada para o comando de API **WSACONTEXT BUILD**. O comando **WSACONTEXT BUILD** usa esta entrada para construir o EPR <wsa:To> para o Serviço de Finalização.
- e. O Cliente de Pedido emite um comando **INVOKE SERVICE** que inicia o processamento de pipeline.
- f. O manipulador de endereçamento de serviços da web do CICS, DFHWSADH, no pipeline de saída converte o EPR <wsa:To> em um endereço e um conjunto opcional de parâmetros de referência que ele coloca no cabeçalho da mensagem de solicitação SOAP que está sendo enviada ao Serviço de Finalização:

```
<env:Header>
  <wsa:To>http://example.ibm.com/Fulfilment_Service</wsa:To>
</env:Header>
```

Figura 28 mostra a solicitação do Cliente de Pedido para o serviço de Finalização. Esta solicitação envolve um pipeline de serviços da web que inclui o manipulador de endereçamento de serviços da web do CICS, DFHWSADH.

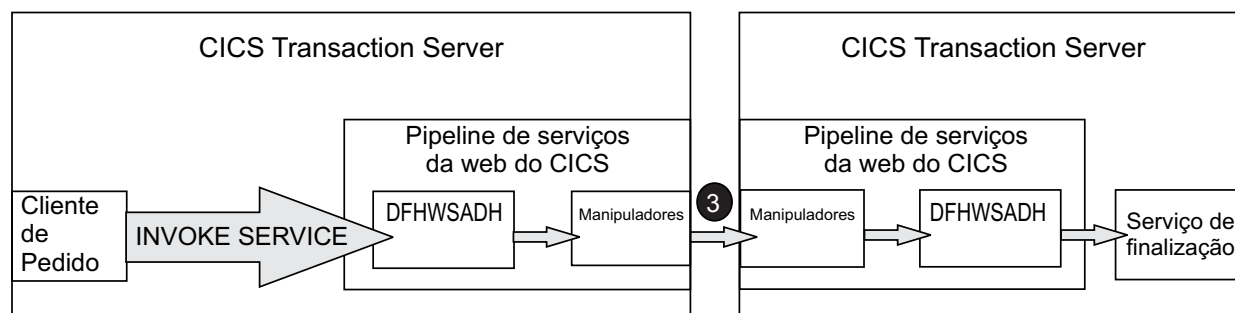


Figura 28. Solicitação do Cliente de Pedido para o Serviço de Finalização

Terminologia do Web Services Addressing

Termos utilizados para explicar o suporte ao Web Services Addressing (WS-Addressing).

contexto de endereçamento

Um documento XML que armazena propriedades de endereçamento de mensagens do WS-Addressing (MAPs) antes que elas sejam enviadas em mensagens de solicitação SOAP e após elas serem recebidas das mensagens de solicitação e resposta SOAP.

referência de terminal (EPR)

Uma estrutura XML contendo informações de endereçamento que são usadas para rotear uma mensagem para um serviço da web. Essas informações de endereçamento incluem o endereço de destino da mensagem, os parâmetros de referência opcionais para uso do aplicativo e metadados opcionais.

propriedade de endereçamento de mensagens (MAP)

Um elemento XML que transmite informações de endereçamento para uma mensagem de serviço da web específica, tal como um ID de mensagem exclusivo, o destino da mensagem e as referências de terminal da mensagem.

Suporte para SAML

O CICS suporta o uso de Security Assertion Markup Language (SAML) para descrever e trocar informações de segurança entre parceiros de negócios on-line.

O CICS suporta os padrões SAMLCore1.1 e SAML Core2.0. Ele não suporta os protocolos que estão descritos nesses padrões.

É possível configurar pipelines do provedor e do solicitante para usar tokens SAML, mas você deve primeiro implementar o CICS Security Token Service (STS). Para obter mais informações sobre como configurar sua instalação do CICS para suportar SAML, consulte Configurando o CICS para SAML.

Capítulo 3. Desenvolvendo serviços da web

Criando um serviço da web JSON

É possível expor os aplicativos CICS existentes como serviços da web JSON e criar novos aplicativos CICS para agir como provedores de serviços da web JSON.

Antes de Iniciar

Antes de começar a criar um serviço da web JSON, deve-se configurar seu sistema CICS para suportar serviços da web JSON. Para obter mais informações, consulte “Criando a infraestrutura do CICS para um provedor de serviços JSON” na página 67.

Sobre Esta Tarefa

O assistente CICS JSON é um utilitário fornecido que o ajuda a criar os artefatos necessários para um novo aplicativo do provedor de serviços da web JSON ou a ativar um aplicativo existente como um provedor de serviço da web JSON.

O assistente CICS JSON pode criar um esquema JSON a partir de uma estrutura de linguagem de alto nível ou de uma estrutura de linguagem de alto nível de um esquema JSON existente; ele suporta COBOL, C/C++ e PL/I. Ele também gera informações que são usadas para ativar a conversão de tempo de execução automática das mensagens JSON para contêineres e COMMAREAs e vice-versa. Essas informações são usadas pelo suporte de serviços da web CICS JSON durante o processamento de pipeline.

Crie seu serviço da web JSON, conforme descrito no procedimento a seguir, e valide se ele funciona corretamente:

Procedimento

1. Crie um serviço da web JSON. Use o assistente JSON para criar o esquema JSON ou as estruturas de linguagem e implementá-los no CICS. Use o comando **PIPELINE SCAN** para criar automaticamente os recursos necessários do CICS.
2. Inicie o serviço da web JSON para testar se ele funciona conforme o desejado.

O que Fazer Depois

Essas etapas são explicadas em mais detalhes nos tópicos a seguir.

O assistente JSON do CICS

O assistente CICS JSON é um conjunto de utilitários em lote que cria um mapeamento entre o esquema JSON e as estruturas de linguagem. Esse mapeamento é usado pelo CICS no tempo de execução para fazer a transformação entre JSON e dados do aplicativo. O assistente suporta a implementação rápida de aplicativos CICS para uso em provedores de serviços e solicitantes de serviços, com o mínimo de esforço de programação.

Quando você usa o assistente JSON para CICS, não é necessário gravar seu próprio código para analisar mensagens de entrada e para construir mensagens de saída; O CICS mapeia dados entre a mensagem JSON e a estrutura de dados do programa de aplicativo.

O assistente pode criar um esquema JSON a partir de uma estrutura de linguagem de alto nível ou uma estrutura de linguagem de alto nível a partir de um esquema JSON existente e suporta COBOL, C/C++ e PL/I. Ele também gera informações utilizadas para ativar a conversão de tempo de execução automática das mensagens JSON para contêineres e COMMAREAs e vice-versa.

O assistente CICS JSON consiste em dois programas utilitários:

DFHLS2JS

Gera um arquivo de ligação de serviço da web a partir de uma estrutura de linguagem. Este utilitário também gera um esquema JSON.

DFHJS2LS

Gera um arquivo de ligação de serviço da web a partir de um esquema JSON. Este utilitário também gera uma estrutura de linguagem que você pode usar em seus programas de aplicativo.

Os procedimentos JCL para executar ambos os programas estão na biblioteca *hlq.XDFHINST*.

O modo de uso relevante para o procedimento DFHLS2JS ou DFHJS2LS depende de seus requisitos:

- DFHLS2JS: linguagem de alto nível para conversão de esquema JSON para a interface vinculável
- DFHJS2LS: esquema JSON para conversão de linguagem de alto nível para interface vinculável
- DFHLS2JS: linguagem de alto nível para conversão de esquema JSON para serviços de solicitação-resposta
- DFHJS2LS: esquema JSON para conversão de linguagem de alto nível para serviços de solicitação-resposta
- DFHJS2LS: Esquema JSON para conversão de linguagem de alto nível para serviços RESTful

Para obter mais informações sobre os programas utilitários e mapeamentos de dados do assistente JSON, consulte os tópicos a seguir.

DFHLS2JS: Linguagem de alto nível para conversão de esquema JSON para serviços de solicitação/resposta

O procedimento DFHLS2JS gera um arquivo de esquema JSON a partir de uma estrutura de dados de linguagem de alto nível. É possível usar o DFHLS2JS ao expor um programa de aplicativo CICS como um provedor de serviços.

As instruções de controle de tarefa para DFHLS2JS, seus parâmetros simbólicos, seus parâmetros de entrada, suas descrições e uma tarefa de exemplo o ajudam a usar este procedimento.

O procedimento JCL DFHLS2JS é instalado no conjunto de dados *HLQ.XDFHINST*, em que *HLQ* é o qualificador de alto nível no qual o CICS está instalado.

Instruções de controle da tarefa para DFHLS2JS

JOB Inicia a tarefa.

EXEC Especifica o nome do procedimento (DFHLS2JS).

INPUT.SYSUT1 DD

Especifica a entrada. Os parâmetros de entrada geralmente são especificados no fluxo de entrada. No entanto, eles podem ser definidos em um conjunto de dados ou em um membro de um conjunto de dados particionados.

Parâmetros simbólicos

Os parâmetros simbólicos a seguir são definidos no DFHLS2JS:

JAVADIR = *path*

Especifica o nome do diretório Java que é usado pelo DFHLS2JS. O valor desse parâmetro é anexado a `/usr/lpp/` para produzir um nome de caminho completo de `/usr/lpp/ path`.

Normalmente, você não especifica este parâmetro. O valor padrão é o valor que foi fornecido à tarefa de instalação do CICS (DFHISTAR) no parâmetro **JAVADIR**.

PATHPREFIX = *prefix*

Especifica um prefixo que estende o caminho do diretório z/OS UNIX que é usado em outros parâmetros ou '' (sequência de caracteres vazia) se nenhum prefixo é usado.

Normalmente, você não especifica este parâmetro. O valor padrão é o valor que foi fornecido para a tarefa de instalação do CICS (DFHISTAR) no parâmetro **PATHPREFIX**.

SERVICE = *value*

Use este parâmetro somente quando orientado a fazer isso pelo Suporte IBM.

TMPDIR = *tmpdir*

Especifica o local de um diretório no z/OS UNIX que o DFHLS2JS usa como uma área de trabalho provisória. O ID do usuário usado para executar a tarefa deve ter permissão de leitura e gravação para esse diretório.

O valor padrão é `/tmp`.

TMPFILE = *tmpprefix*

Especifica um prefixo que o DFHLS2JS utiliza para construir os nomes dos arquivos de área de trabalho provisória.

O valor padrão é `LS2JS`.

USSDIR = *path*

Especifica o nome do diretório TS do CICS no sistema de arquivos do UNIX System Services. O valor desse parâmetro é anexado a `/usr/lpp/cicsts/` para produzir um nome do caminho completo de `/usr/lpp/cicsts/ path`. Isso deve ser especificado como `'.'` (ponto) se o padrão for usado.

Normalmente, você não especifica este parâmetro. O valor padrão é o valor que foi fornecido à tarefa de instalação do CICS (DFHISTAR) no parâmetro **USSDIR**.

A área de trabalho provisória

DFHLS2JS cria os três arquivos temporários a seguir no tempo de execução:

```
tmpdir / tmpprefix .in  
tmpdir / tmpprefix .out  
tmpdir / tmpprefix .err
```

em que:

tmpdir é o valor especificado no parâmetro **TMPDIR**.

tmpprefix é o valor especificado no parâmetro **TMPFILE**.

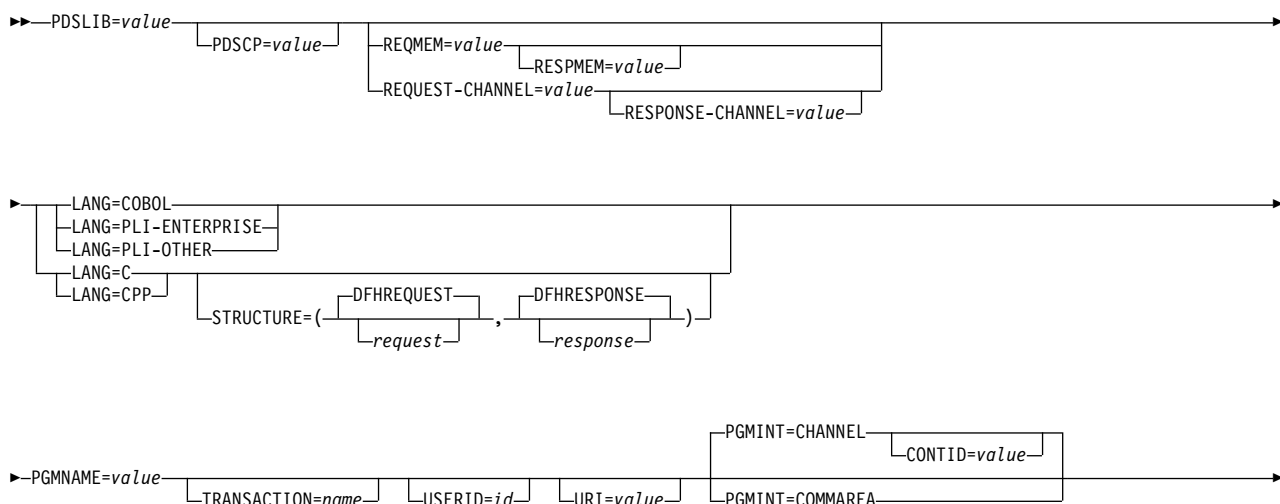
Os nomes padrão para os arquivos quando **TMPDIR** e **TMPFILE** não são especificados, são conforme a seguir:

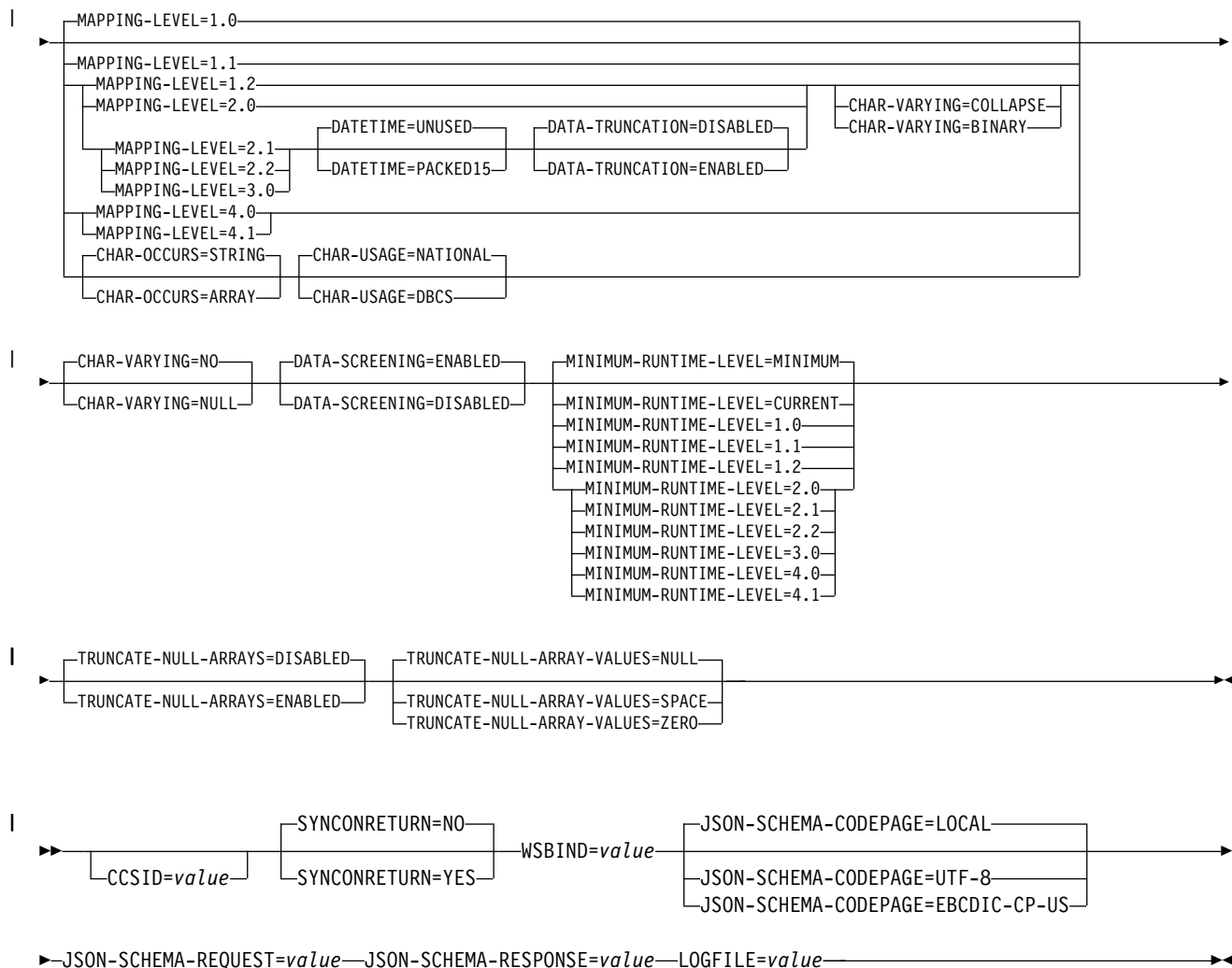
```
/tmp/LS2JS.in  
/tmp/LS2JS.out  
/tmp/LS2JS.err
```

Importante: O DFHLS2JS não bloqueia o acesso aos arquivos z/OS UNIX ou aos membros do conjunto de dados. Portanto, se duas ou mais instâncias de DFHLS2JS forem executadas simultaneamente e usarem os mesmos arquivos de área de trabalho provisória, nada impede que uma tarefa sobrescreva os arquivos da área de trabalho enquanto outra tarefa está usando-os, levando a falhas imprevisíveis.

Portanto, você será avisado para planejar uma convenção de nomenclatura e procedimentos operacionais que evitam essa situação. Por exemplo, é possível usar o parâmetro simbólico do sistema **SYSUID** para gerar nomes de arquivos de área de trabalho que são exclusivos para um usuário individual. Estes arquivos temporários são excluídos antes do encerramento da tarefa.

Parâmetros de entrada para DFHLS2JS





Utilização de Parâmetros

- Você pode especificar os parâmetros de entrada em qualquer ordem.
- Cada parâmetro deve ser iniciado em uma nova linha.
- Um parâmetro, e seu caractere de continuação, se você usar um, não deve estender além da coluna 72; as colunas 73 a 80 devem conter espaços em branco.
- Se um parâmetro é muito longo para se ajustar em uma única linha, use um caractere de asterisco (*) no final da linha para indicar que o parâmetro continua na próxima linha. Tudo, incluindo espaços antes do asterisco, é considerado parte do parâmetro. Por exemplo:

```
WSBIND=wsbinddir*
/app1
```

é equivalente a

```
WSBIND=wsbinddir/app1
```

- Um caractere # na primeira posição de caractere da linha é um caractere de comentário. A linha é ignorada.
- Uma vírgula na posição de último caractere da linha é um separador de linha opcional e é ignorado.

Descrições de Parâmetros

CCSID = *value*

Especifica o CCSID que é usado no tempo de execução para codificar dados de caractere na estrutura de dados do aplicativo. O valor desse parâmetro substitui o valor do parâmetro de inicialização do sistema **LOCALCCSID**. O *value* deve ser um CCSID EBCDIC suportado por Java e . Se você não especificar esse parâmetro, a estrutura de dados do aplicativo será codificada usando o CCSID especificado no parâmetro de inicialização do sistema.

CHAR-VARYING = { **NO** | **NULL** | **COLLAPSE** | **BINARY** }

Especifica como os campos de caractere na estrutura de linguagem são mapeados quando o nível de mapeamento é 1.2 ou superior. Um campo de caractere em COBOL é uma cláusula Picture do tipo X, por exemplo PIC(X) 10; um campo de caractere em C/C++ é uma matriz de caracteres. É possível selecionar essas opções:

NO Campos de caractere são mapeados para uma sequência JSON e são processados como campos de comprimento fixo. O comprimento máximo dos dados é igual ao comprimento do campo. NO é o valor padrão para o parâmetro **CHAR-VARYING** para COBOL e PL/I nos níveis de mapeamento 2.0 e anterior.

Esse valor não se aplica às estruturas de linguagem Enterprise e Outro PL/I.

NULL Os campos de caractere são mapeados para uma sequência JSON e são processados como sequências com terminação nula. O CICS inclui um caractere com terminação nula ao transformar a partir de uma mensagem JSON. O comprimento máximo da sequência de caracteres é calculado como um caractere menor que o comprimento indicado na estrutura de linguagem. NULL é o valor padrão para o parâmetro **CHAR-VARYING** para C/C++.

Esse valor não se aplica às estruturas de linguagem Enterprise e Outro PL/I.

COLLAPSE

Os campos de caractere são mapeados para uma sequência JSON. Espaços em branco finais no campo não são incluídos na mensagem JSON. A mensagem JSON recebida é analisada para remover todo espaço em branco inicial, final e integrado. COLLAPSE é o valor padrão para o parâmetro **CHAR-VARYING** para COBOL e PL/I no nível de mapeamento 2.1 em diante.

BINARY

Os campos de caractere são mapeados para uma sequência JSON contendo dados codificados em base64 e são processados como campos de comprimento fixo. O valor BINARY no parâmetro **CHAR-VARYING** está disponível somente no nível de mapeamento 2.1 e posteriores.

CHAR-OCCURS = { **STRING** | **ARRAY** }

Especifica como matrizes de caracteres na estrutura de linguagem são mapeadas quando o nível de mapeamento é 4.0 ou superior. Por exemplo, PIC X OCCURS 20. Este parâmetro destina-se ao uso somente pela linguagem COBOL.

ARRAY

As matrizes de caracteres são mapeadas para uma matriz JSON. Isso

significa que cada caractere é mapeado como um elemento JSON individual. Este também é o comportamento nos níveis de mapeamento 3.0 e anteriores.

CADEIA

Matrizes de caracteres são mapeadas para uma sequência JSON. Isso significa que a matriz COBOL inteira é mapeada como um único elemento JSON.

CHAR-USAGE = { NATIONAL | DBCS }

Em COBOL, o tipo de dado nacional, PIC N, pode ser usado para dados UTF-16 ou DBCS. Essa configuração é controlada pela opção do compilador NSYMBOL. Deve-se configurar o parâmetro **CHAR-USAGE** no assistente com o mesmo valor que a opção do compilador NSYMBOL para assegurar que os dados sejam manipulados apropriadamente. Isso geralmente é configurado como CHAR-USAGE=NATIONAL quando você usa UTF-16.

DBCS Dados dos campos PIC (*n*) são tratados como dados codificados em DBCS.

NATIONAL

Dados dos campos PIC (*n*) são tratados como dados codificados em UTF-16.

CONTID = *value*

Em um provedor de serviços, especifica o nome do contêiner que contém a estrutura de dados de nível superior que é usada para representar uma mensagem JSON.

O comprimento do contêiner que o CICS transmite para o programa de aplicativo de destino é o maior dos comprimentos do contêiner de solicitação e do contêiner de resposta.

DATA-SCREENING = { ENABLED | DISABLED }

Especifica se os dados fornecidos pelo aplicativo foram verificados com relação aos erros.

ENABLED

Quaisquer dados inconsistentes de tempo de execução fornecidos pelo aplicativo com a estrutura de linguagem são tratados como um erro e a mensagem DFHPI1010 é emitida. Uma resposta de erro é retornada ao aplicativo.

DISABLED

Os valores dos dados de tempo de execução fornecidos pelo aplicativo que são inconsistentes com a estrutura de linguagem são substituídos pelos valores padrão. Por exemplo, um zero substitui um valor inválido em um campo numérico. A mensagem DFHPI1010 não é emitida e uma resposta normal é retornada ao aplicativo. Esse recurso pode ser usado para evitar as respostas de erro INVALID_PACKED_DEC e INVALID_ZONED_DEC que são geradas a partir de campos de saída não inicializados.

DATA-TRUNCATION = { DISABLED | ENABLED }

Especifica se dados de comprimento variável são tolerados em uma estrutura de campo de comprimento fixo:

DISABLED

Se os dados forem menores do que o comprimento fixo que o CICS está esperando, o CICS rejeitará os dados truncados e emitirá uma mensagem de erro.

ENABLED

Se os dados são menores do que o comprimento fixo que o CICS está esperando, o CICS tolera os dados truncados e processa os dados ausentes como valores nulos.

DATETIME = { **UNUSED** | **PACKED15** }

Especifica se campos ABSTIME em potencial na estrutura de linguagem de alto nível são mapeados como registros de data e hora:

PACKED15

Campos decimais compactados de comprimento 15 (8 bytes) são tratados como campos CICS ABSTIME e mapeados como registros de data e hora.

UNUSED

Campos decimais compactados de comprimento 15 (8 bytes) não são tratados como registros de data e hora.

Você pode configurar este parâmetro no nível de mapeamento 3.0.

JSON-SCHEMA-CODEPAGE = { **LOCAL** | **UTF-8** | **EBCDIC-CP-US** }

Especifica a página de códigos que é usada para gerar os documentos do Esquema JSON.

LOCAL

Especifica que os Esquemas JSON são gerados usando a página de códigos padrão para o sistema de arquivos.

UTF-8 Especifica que os Esquemas JSON são gerados usando a página de códigos UTF-8.

EBCDIC-CP-US

Especifica que os Esquemas JSON são gerados usando a página de códigos US EBCDIC.

JSON-SCHEMA-REQUEST = *value*

Este é um parâmetro obrigatório.

O valor indica o local do UNIX System Services no qual o esquema JSON da solicitação é armazenado.

JSON-SCHEMA-RESPONSE = *value*

Este é um parâmetro obrigatório.

O valor indica o local do UNIX System Services no qual o esquema JSON de resposta é armazenado.

LANG = **COBOL**

Especifica que a linguagem de programação da estrutura de linguagem de alto nível é COBOL.

LANG = **PLI-ENTERPRISE**

Especifica que a linguagem de programação da estrutura de linguagem de alto nível é Enterprise PL/I.

LANG = **PLI-OTHER**

Especifica que a linguagem de programação da estrutura de linguagem de alto nível é um nível de PL/I diferente de Enterprise PL/I.

LANG = **C**

Especifica que a linguagem de programação da estrutura de linguagem de alto nível é C.

LANG = CPP

Especifica que a linguagem de programação da estrutura de linguagem de alto nível é C++.

LOGFILE = value

O nome completo do arquivo z/OS UNIX no qual o DFHLS2JS grava seu log de atividades e informações de rastreamento. DFHLS2JS cria o arquivo, mas não a estrutura de diretório, se ele não existe.

Geralmente, você não usa esse arquivo, mas ele poderá ser solicitado pela organização de serviço da IBM caso você encontre problemas com o DFHLS2JS.

MAPPING-LEVEL = { 1.0 | 1.1 | 1.2 | 2.0 | 2.1 | 2.2 | 3.0 | 4.0 | 4.1 }

Especifica o nível de mapeamento que o DFHLS2JS usa quando você gera o arquivo de ligação de serviço da web e o esquema JSON. Você pode selecionar estas opções:

- 1.0** Essa opção é mantida para compatibilidade com serviços da web SOAP. Ela não é recomendada para uso com JSON.
- 1.1** Essa opção é mantida para compatibilidade com serviços da web SOAP. Ela não é recomendada para uso com JSON.
- 1.2** Essa opção é mantida para compatibilidade com serviços da web SOAP. Ela não é recomendada para uso com JSON.
- 2.0** Essa opção é mantida para compatibilidade com serviços da web SOAP. Ela não é recomendada para uso com JSON.
- 2.1** Essa opção é mantida para compatibilidade com serviços da web SOAP. Ela não é recomendada para uso com JSON.
- 2.2** Essa opção é mantida para compatibilidade com serviços da web SOAP. Ela não é recomendada para uso com JSON.
- 3.0** Use este nível de mapeamento para gerar o esquema JSON usando o conjunto completo de opções disponíveis.
- 4.0** Use este nível de mapeamento com uma região do CICS TS 5.2 ou mais recente. Neste nível de mapeamento é possível usar campos OCCURS DEPENDING ON de COBOL e o parâmetro **CHAR-OCCURS**.
- 4.1** Use este nível de mapeamento para o suporte da matriz truncável, com uma região CICS TS V5.2 que tem o APAR PI67641 aplicado ou uma região CICS TS V5.3 ou mais recente.

Para obter mais informações sobre níveis de mapeamento, consulte Níveis de mapeamento para os assistentes CICS JSON.

MINIMUM-RUNTIME-LEVEL = { MINIMUM | 1.0 | 1.1 | 1.2 | 2.0 | 2.1 | 2.2 | 3.0 | 4.0 | 4.1 | CURRENT }

Especifica o ambiente de tempo de execução mínimo do CICS no qual o arquivo de ligação de serviço da web pode ser implementado. Se você selecionar um nível que não corresponde aos outros parâmetros que você especificou, receberá uma mensagem de erro. É possível selecionar essas opções:

MINIMUM

O menor nível de tempo de execução do CICS possível é alocado automaticamente conforme os parâmetros que você selecionou.

- 1.0** Essa opção é mantida para compatibilidade com serviços da web SOAP. Ela não é recomendada para uso com JSON.

- 1.1 Essa opção é mantida para compatibilidade com serviços da web SOAP. Ela não é recomendada para uso com JSON.
- 1.2 Essa opção é mantida para compatibilidade com serviços da web SOAP. Ela não é recomendada para uso com JSON.
- 2.0 Essa opção é mantida para compatibilidade com serviços da web SOAP. Ela não é recomendada para uso com JSON.
- 2.1 Essa opção é mantida para compatibilidade com serviços da web SOAP. Ela não é recomendada para uso com JSON.
- 2.2 Essa opção é mantida para compatibilidade com serviços da web SOAP. Ela não é recomendada para uso com JSON.
- 3,0 O arquivo de ligação de serviço da web gerado é implementado em uma região do CICS TS 4.1 ou mais recente.

Nota: O suporte JSON está disponível somente a partir do CICS TS 4.2 em diante.

- 4.0 O arquivo de ligação de serviço da web gerado é implementado com sucesso em um CICS TS 5.2 ou em uma região posterior. Com este nível de tempo de execução, é possível usar um nível de mapeamento 4.0 ou anterior para o parâmetro **MAPPING-LEVEL**. É possível utilizar qualquer parâmetro opcional neste nível.
- 4.1 O arquivo de ligação de serviço da web gerado é implementado com sucesso em uma região CICS TS V5.2 que tem o APAR PI67641 aplicado ou em qualquer região CICS TS V5.3 ou mais recente. Com este nível de tempo de execução, é possível utilizar um nível de mapeamento 4.1 ou anterior para o parâmetro **MAPPING-LEVEL**.

CURRENT

O arquivo de ligação de serviço da web gerado é implementado com sucesso em uma região CICS no mesmo nível de tempo de execução que você está usando para gerar o arquivo de ligação de serviço da web.

PDSLIB = *value*

Especifica o nome do conjunto de dados particionados que contém as estruturas de dados de linguagem de alto nível a serem processadas. Os membros do conjunto de dados que são usados para a solicitação e resposta são especificados nos parâmetros **REQMEM** e **RESPMEM**.

Restrição: Os registros no conjunto de dados particionados deve ter um comprimento fixo de 80 bytes.

PDSCP = *value*

Especifica a página de códigos que é usada nos membros do conjunto de dados particionados que são especificados nos parâmetros **REQMEM** e **RESPMEM**, em que *value* é um número de CCSID ou um número da página de códigos Java. Se este parâmetro não for especificado, a página de códigos do z/OS UNIX System Services será usada. Por exemplo, você pode especificar **PDSCP** = 037.

PGMINT = { **CHANNEL** | **COMMAREA** }

Para um provedor de serviços, especifica como o CICS transmite dados ao programa de aplicativo de destino:

CHANNEL

CICS usa uma interface do canal para transmitir dados ao programa de aplicativo de destino.

- Em níveis de mapeamento anteriores a 3.0, o canal pode conter somente um contêiner, que é usado para entrada e saída. Use o parâmetro **CONTID** para especificar o nome do contêiner. O nome padrão é DFHWS-DATA.
- No nível de mapeamento 3.0, o canal pode conter vários contêineres. Use os parâmetros **REQUEST-CHANNEL** e **RESPONSE-CHANNEL**. Não especifique **PDSLIB**, **REQMEM** ou **RESPMEM**.

COMMAREA

CICS usa uma área de comunicação para transmitir dados ao programa de aplicativo de destino.

Quando o programa de aplicativo de destino tiver processado a solicitação, ele deverá usar o mesmo mecanismo para retornar a resposta. Se a solicitação foi recebida em uma área de comunicação, a resposta deverá ser retornada na área de comunicação; se a solicitação foi recebida em um contêiner, a resposta deverá ser retornada em um contêiner. O comprimento da área de comunicação ou contêiner que o CICS transmite para o programa de aplicativo de destino é o maior dos comprimentos da área ou do contêiner de comunicação da solicitação e da área ou contêiner de comunicação de resposta.

PGMNAME = *value*

Especifica o nome do recurso PROGRAM do CICS para o programa de aplicativo de destino que é exposto como um serviço da web. O suporte de serviço da web do CICS está vinculado a este programa.

REQMEM = *value*

Especifica o nome do membro do conjunto de dados particionados que contém a estrutura de linguagem de alto nível para a solicitação de serviço da web. Para um provedor de serviços, a solicitação de serviço da web é a entrada para o programa de aplicativo.

REQUEST-CHANNEL = *value*

Especifica o nome e o local de um documento de descrição de canal. A descrição de canal descreve os contêineres que o aplicativo do provedor de serviço da web pode usar em sua interface quando ele recebe uma mensagem JSON de um solicitante de serviço da web. A descrição de canal é um documento XML que deve estar em conformidade com o esquema de canal fornecido pelo CICS. Para obter mais informações, consulte “Criando um Documento de Descrição de Canal” na página 276.

Você pode usar esse parâmetro somente no nível de mapeamento 3.0.

RESPMEM = *value*

Especifica o nome do membro do conjunto de dados particionados que contém a estrutura de linguagem de alto nível para a resposta de serviço da web. Para um provedor de serviços, a resposta de serviço da web é a saída do programa de aplicativo.

RESPONSE-CHANNEL = *value*

Especifica o nome e o local de um documento de descrição de canal. A descrição de canal descreve os contêineres que o aplicativo do provedor de serviço da web pode usar em sua interface quando ele envia uma mensagem de resposta JSON para um solicitante de serviço da web. A descrição de canal é um documento XML que deve estar em conformidade com o esquema de

canal fornecido pelo CICS. Para obter mais informações, consulte “Criando um Documento de Descrição de Canal” na página 276.

Você pode usar esse parâmetro somente no nível de mapeamento 3.0.

STRUCTURE = (*request* , *response*)

Apenas para C e C++, especifica os nomes das estruturas de alto nível que estão contidas nos membros do conjunto de dados particionados que são especificados nos parâmetros **REQMEM** e **RESPMEM**:

request

Especifica o nome da estrutura de alto nível que contém a solicitação quando o parâmetro **REQMEM** é especificado. O valor padrão é DFHREQUEST.

O membro do conjunto de dados particionados deve conter uma estrutura de alto nível com o nome que você especifica ou uma estrutura denominada DFHREQUEST se você não especificar um nome.

response

Especifica o nome da estrutura de alto nível que contém a resposta quando o parâmetro **RESPMEM** é especificado. O valor padrão é DFHRESPONSE.

Se você especificar um valor, o membro do conjunto de dados particionados deverá conter uma estrutura de alto nível com o nome que você especificar ou uma estrutura denominada DFHRESPONSE se você não especificar um nome.

SYNCONRETURN = { NO | **YES** }

Especifica se o serviço da web remoto pode emitir um ponto de sincronização.

NO O serviço da web remoto não pode emitir um ponto de sincronização. Este valor é o padrão. Se o serviço da web remoto emitir um ponto de sincronização, ele falhará com um encerramento anormal de ADPL.

YES O serviço da web remoto pode emitir um ponto de sincronização. Se você selecionar YES, a tarefa remota é confirmada como uma unidade de trabalho separada quando o controle retorna do serviço da web remoto. Se o serviço da web remoto atualizar um recurso recuperável e ocorrer uma falha após ele retornar, a atualização para esse recurso não poderá ser restaurada.

TRANSACTION = *name*

Em um provedor de serviços, este parâmetro especifica um nome com um a quatro caracteres de uma transação de alias que pode iniciar o pipeline. O valor desse parâmetro é usado para definir o atributo TRANSACTION do recurso URIMAP quando ele é criado automaticamente usando o comando de varredura **PIPELINE**.

Caracteres aceitáveis:

A-Z a-z 0-9 \$ @ # _ < >

TRUNCATE-NULL-ARRAYS = { **DISABLED** | **ENABLED** }

Especifica como matrizes estruturadas são processadas no nível de mapeamento 4.1 ou superior. Se ativado, o CICS tentará reconhecer registros vazios em uma matriz (consulte TRUNCATE-NULL-ARRAY-VALUES para obter mais informações sobre como identificar os registros vazios). Se cinco registros de matriz vazia consecutivos forem detectados, a matriz será truncada no primeiro registro ao gerar XML/JSON. Esse recurso de truncamento é ativado somente para matrizes com conteúdo estruturado, matrizes de campos

primitivos simples não estão sujeitas a truncamento. O truncamento de matrizes pode resultar em uma representação mais concisa dos dados em JSON/XML, mas não é sem risco. Se cinco registros de dados consecutivos forem identificados incorretamente como armazenamento não inicializado (talvez porque eles legitimamente contêm valores baixos), poderá haver perda de dados. Se TRUNCATE-NULL-ARRAYS estiver ativado e TRUNCATE-NULL-ARRAY-VALUES não estiver configurado, o valor padrão para TRUNCATE-NULL-ARRAY-VALUES será usado.

TRUNCATE-NULL-ARRAY-VALUES = { NULL | **SPACE** | **ZERO** }

Especifica quais valores são tratados como vazios para processamento de TRUNCATE-NULL-ARRAYS no nível de mapeamento 4.1 ou superior. Por padrão, o valor nulo (0x00 ou valores baixos) é tratado como vazio. Se todos os bytes de armazenamento em um registro de uma matriz estruturada contêm valores nulos, o registro inteiro é considerado vazio. Um ou mais dos valores NULL, SPACE e ZERO podem ser especificados em uma lista separada por vírgula, em que NULL indica um caractere nulo (0x00), SPACE indica um espaço SBCS EBCDIC (0x40) e ZERO indica um zero decimal zonado não sinalizado (0xF0). Qualquer combinação correspondente dos bytes selecionados em um registro de matriz estruturada fará com que o registro inteiro seja identificado como vazio. Se TRUNCATE-NULL-ARRAY-VALUES tem um valor definido, TRUNCATE-NULL-ARRAYS deve ser ativado.

URI = *value*

Este parâmetro especifica o URI relativo ou absoluto que um cliente usa para acessar o serviço da web. O CICS usa o valor que é especificado quando ele gera um recurso URIMAP a partir do arquivo de ligação de serviço da web que é criado por DFHLS2JS. O parâmetro especifica o componente de caminho do URI ao qual a definição de URIMAP se aplica.

USERID = *id*

Em um provedor de serviços, este parâmetro especifica um ID do usuário de um a oito caracteres, o qual pode ser usado por qualquer Web client. Para uma resposta gerada por aplicativo ou um serviço da web, a transação de alias é conectada com este ID do usuário. O valor desse parâmetro é usado para definir o atributo USERID do recurso URIMAP quando ele é criado automaticamente usando o comando de varredura **PIPELINE**.

Caracteres aceitáveis:

A-Z a-z 0-9 \$ @ #

WSBIND = *value*

O nome completo do z/OS UNIX do arquivo de ligação de serviço da web. O DFHLS2JS cria o arquivo, mas não a estrutura de diretório, se ele não existe. A extensão do arquivo é .wsbind.

Outras informações

- O ID do usuário que o DFHLS2JS usa para execução deve ser configurado para usar o UNIX System Services. O ID do usuário deve ter permissão de leitura para a estrutura do arquivo CICS z/OS UNIX e bibliotecas PDS e permissão de gravação para os diretórios especificados nos parâmetros **LOGFILE**, **WSBIND** e **JSON Schema**.
- O ID do usuário deve ter uma alocação de armazenamento suficientemente grande para executar Java.

- A JCL possui um comprimento de parâmetro máximo de 100 caracteres. Este comprimento de parâmetro pode ser aumentado usando a instrução **STDPARM**, para obter mais informações, consulte *z/OS UNIX System Services User Guide*.

Exemplo

```
//LS2JS JOB '
accounting information
',
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: Esquema JSON para conversão de linguagem de alto nível para serviços de solicitação/resposta

O procedimento DFHJS2LS gera uma estrutura de dados de linguagem de alto nível e um arquivo de ligação de serviço da web a partir de um esquema JSON. É possível usar DFHJS2LS ao se preparar para criar um programa de aplicativo CICS como um provedor de serviços. Este tópico lista as instruções de controle de tarefa, os parâmetros simbólicos, os parâmetros de entrada e suas descrições para DFHJS2LS.

O procedimento JCL DFHJS2LS é instalado no conjunto de dados *HLQ.XDFHINST*, em que *HLQ* é o qualificador de alto nível no qual o CICS está instalado.

Instruções de controle da tarefa para DFHJS2LS

JOB Inicia a tarefa.

EXEC Especifica o nome do procedimento (DFHJS2LS).

INPUT.SYSUT1 DD

Especifica a entrada. Os parâmetros de entrada geralmente são especificados no fluxo de entrada. No entanto, eles podem ser definidos em um conjunto de dados ou em um membro de um conjunto de dados particionados.

Parâmetros simbólicos

Os parâmetros simbólicos a seguir são definidos no DFHJS2LS:

JAVADIR = *path*

Especifica o nome do diretório Java que é usado pelo DFHJS2LS. O valor desse parâmetro é anexado a */usr/lpp/* para produzir um nome de caminho completo de */usr/lpp/ path*.

Normalmente, você não especifica este parâmetro. O valor padrão é o valor que foi fornecido à tarefa de instalação do CICS (DFHISTAR) no parâmetro **JAVADIR**.

PATHPREF = *prefix*

Especifica um prefixo que estende o caminho do diretório z/OS UNIX que é usado em outros parâmetros ou '' (sequência de caracteres vazia) se nenhum prefixo é usado.

Normalmente, você não especifica este parâmetro. O valor padrão é o valor que foi fornecido para a tarefa de instalação do CICS (DFHISTAR) no parâmetro **PATHPREF**.

SERVICE = *value*

Use este parâmetro somente quando orientado a fazer isso pelo Suporte IBM.

TMPDIR = *tmpdir*

Especifica o local de um diretório no z/OS UNIX que o DFHJS2LS utiliza como uma área de trabalho provisória. O ID do usuário sob o qual a tarefa é executada deve ter permissão de leitura e gravação para esse diretório.

O valor padrão é /tmp.

TMPFILE = *tmpprefix*

Especifica um prefixo que o DFHJS2LS usa para construir os nomes dos arquivos de área de trabalho provisória.

O valor padrão é JS2LS.

USSDIR = *path*

Especifica o nome do diretório TS do CICS no sistema de arquivos do UNIX System Services. O valor desse parâmetro é anexado a /usr/lpp/cicsts/ para produzir um nome do caminho completo de /usr/lpp/cicsts/ *path* . Isso deve ser especificado como '.' (ponto) se o padrão for usado.

Normalmente, você não especifica este parâmetro. O valor padrão é o valor que foi fornecido à tarefa de instalação do CICS (DFHISTAR) no parâmetro **USSDIR**.

A área de trabalho provisória

DFHJS2LS cria os três arquivos temporários a seguir no tempo de execução:

```
tmpdir / tmpprefix .in  
tmpdir / tmpprefix .out  
tmpdir / tmpprefix .err
```

em que:

tmpdir é o valor especificado no parâmetro **TMPDIR**.

tmpprefix é o valor especificado no parâmetro **TMPFILE**.

Os nomes padrão para os arquivos quando **TMPDIR** e **TMPFILE** não são especificados, são conforme a seguir:

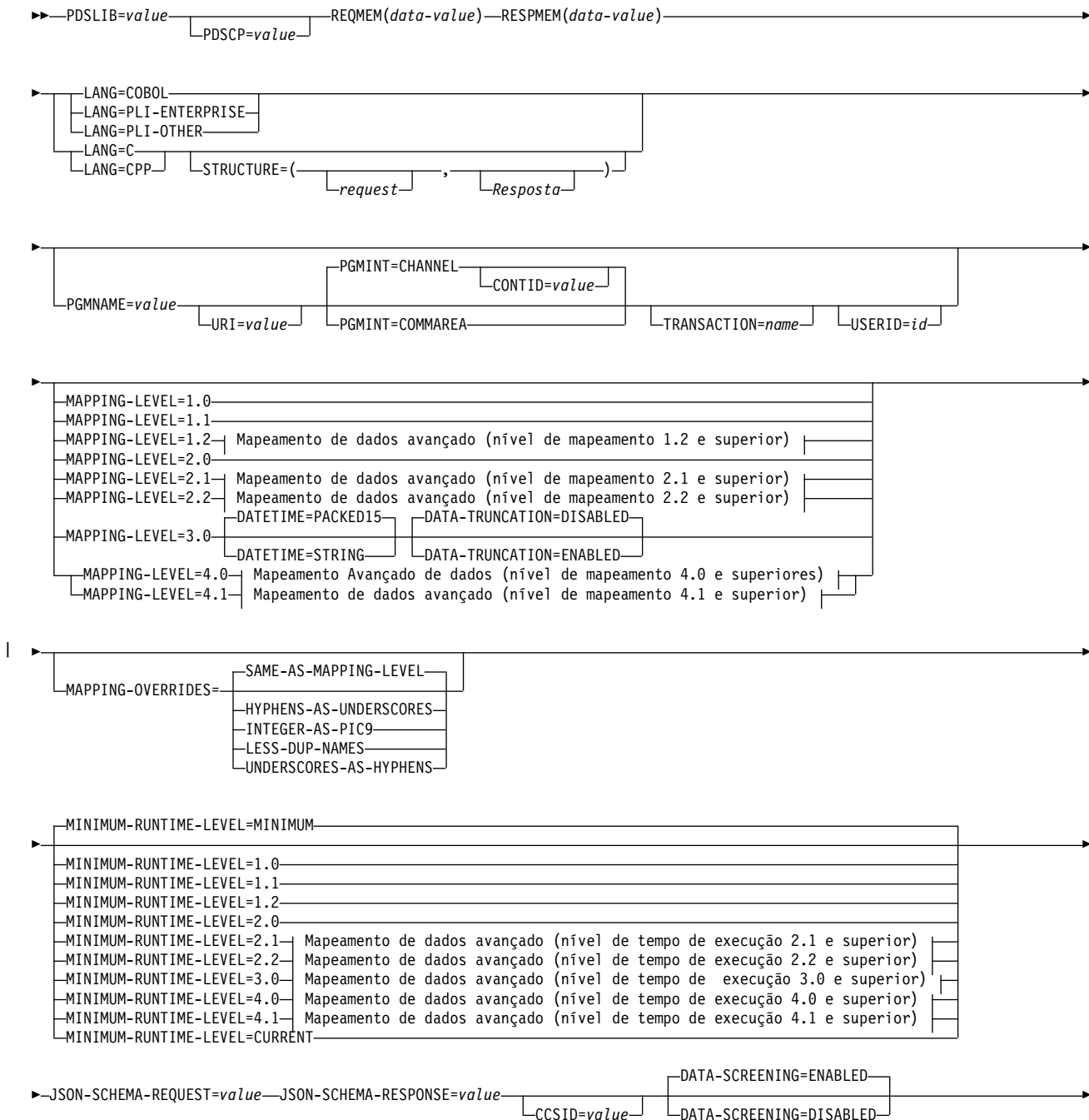
```
/tmp/JS2LS.in  
/tmp/JS2LS.out  
/tmp/JS2LS.err
```

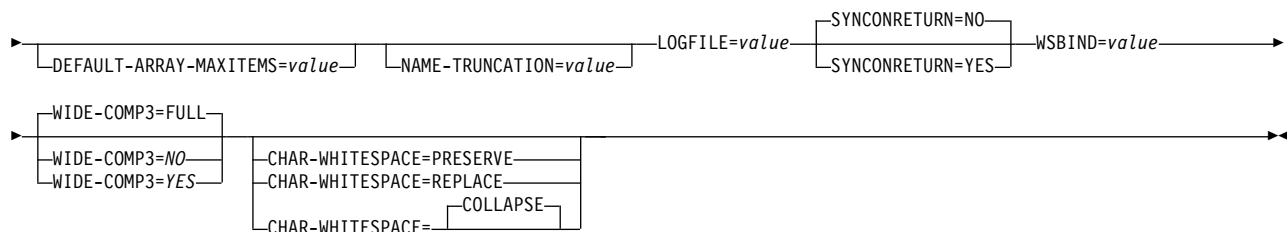
Importante: O DFHJS2LS não bloqueia o acesso aos arquivos do z/OS UNIX ou aos membros do conjunto de dados. Portanto, se duas ou mais instâncias de DFHJS2LS são executadas simultaneamente e usam os mesmos arquivos da área de

trabalho provisória, nada impede que uma tarefa sobrescreva os arquivos da área de trabalho enquanto outra tarefa está usando-os, levando a falhas imprevisíveis.

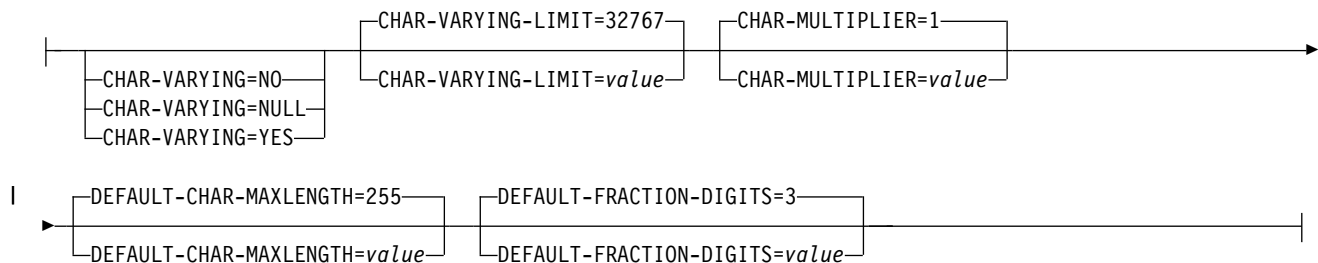
Portanto, você será avisado para planejar uma convenção de nomenclatura e procedimentos operacionais que evitam essa situação. Por exemplo, é possível usar o parâmetro simbólico do sistema **SYSUID** para gerar nomes de arquivos de área de trabalho que são exclusivos para um usuário individual. Estes arquivos temporários são excluídos antes do encerramento da tarefa.

Parâmetros de entrada para DFHJS2LS





Mapeamento de Dados Avançado (Nível de Mapeamento 1.2 e Superior):



Mapeamento de Dados Avançado (Nível de Mapeamento 2.1 e Superior):



Utilização de Parâmetros

- Você pode especificar os parâmetros de entrada em qualquer ordem.
- Cada parâmetro deve ser iniciado em uma nova linha.
- Um parâmetro, e seu caractere de continuação, se você usar um, não deve estender além da coluna 72; as colunas 73 a 80 devem conter espaços em branco.
- Se um parâmetro é muito longo para se ajustar em uma única linha, use um caractere de asterisco (*) no final da linha para indicar que o parâmetro continua na próxima linha. Tudo, incluindo espaços antes do asterisco, é considerado parte do parâmetro. Por exemplo:

```
WSBIND=wsbinddir*
/app1
```

é equivalente a

```
WSBIND=wsbindir/app1
```

- Um caractere # na primeira posição de caractere da linha é um caractere de comentário. A linha é ignorada.
- Uma vírgula na posição de último caractere da linha é um separador de linha opcional e é ignorado.

Descrições de Parâmetros

CCSID = *value*

Especifica o CCSID que é usado no tempo de execução para codificar dados de caractere na estrutura de dados do aplicativo. O valor desse parâmetro substitui o valor do parâmetro de inicialização do sistema **LOCALCCSID**. O *value* deve ser um CCSID EBCDIC suportado por Java e . Se você não especificar

esse parâmetro, a estrutura de dados do aplicativo será codificada usando o CCSID especificado no parâmetro de inicialização do sistema.

CHAR-MULTIPLIER = { 1 | *value* }

Especifica o número de bytes a ser permitido para cada caractere quando o nível de mapeamento é 1.2 ou mais recente. O *value* deste parâmetro pode ser um número inteiro positivo no intervalo de 1 a 2.147.483.647. Todos os mapeamentos baseados em caracteres não numéricos estão sujeitos a este multiplicador. Campos binários, numéricos, zoneados e decimais compactados não estão sujeitos a este multiplicador.

Este parâmetro pode ser útil se, por exemplo, você está planejando usar caracteres DBCS onde pode optar por um multiplicador de 3 para permitir espaço para possíveis caracteres shift-out e shift-in em cada caractere de byte duplo no tempo de execução.

Quando você configura **CCSID=1200** (indicando UTF-16), os únicos valores válidos para **CHAR-MULTIPLIER** são 2 ou 4. Ao usar UTF-16, o valor padrão é 2. Use **CHAR-MULTIPLIER=2** quando esperar que os dados do aplicativo contenham caracteres que requerem 1 unidade de codificação UTF-16. Use **CHAR-MULTIPLIER=4** quando esperar que os dados do aplicativo contenham caracteres que requerem 2 unidades de codificação UTF-16.

Nota: A configuração de **CHAR-MULTIPLIER** como 1 não impede o uso de caracteres DBCS e sua configuração como 2 não impede o uso de pares substitutos UTF-16. No entanto, se caracteres largos forem usados rotineiramente, alguns valores válidos não se ajustarão no campo alocado. Se um valor **CHAR-MULTIPLIER** maior for usado, poderá ser possível armazenar mais caracteres no campo alocado que são válidos no XML. Deve-se tomar cuidado com relação à conformidade com as restrições de intervalo apropriadas.

CHAR-VARYING = { **NO** | **NULL** | **YES** }

Especifica como os dados de caracteres de comprimento variável são mapeados quando o nível de mapeamento é 1.2 ou superior. Os tipos de dados binários de comprimento variável são sempre mapeados para um contêiner ou uma estrutura variável. Se você não especificar esse parâmetro, o mapeamento padrão dependerá da linguagem especificada. Você pode selecionar estas opções:

NO Os dados de caractere de comprimento variável são mapeados como sequências de comprimento fixo.

NULL Os dados de caractere de comprimento variável são mapeados para sequências com terminação nula.

YES Dados de caractere de comprimento variável são mapeados para um tipo de dados CHAR VARYING em PL/I. Nas linguagens COBOL, C e C++, os dados de caracteres de comprimento variável são mapeados para uma representação equivalente que consiste em dois elementos relacionados: dados de comprimento e dados.

CHAR-VARYING-LIMIT = { 32767 | *value* }

Especifica o tamanho máximo de dados binários e de dados de caractere de comprimento variável que são mapeados para a estrutura de linguagem quando o nível de mapeamento é 1.2 ou superior. Se os dados de caractere ou binários forem maiores do que o valor especificado neste parâmetro, eles serão mapeados para um contêiner e o nome do contêiner será usado na estrutura de linguagem gerada. O valor pode variar de 0 até o padrão de 32.767 bytes.

CHAR-WHITESPACE = **COLLAPSE** | **REPLACE** | **PRESERVE**

Especifica como o espaço em branco nos valores de sequência de tipos será manipulado pelo CICS.

COLLAPSE

Espaços em branco iniciais, finais e integrados são removidos e todas as guias, novas linhas e espaços consecutivos são substituídos por caracteres de espaço único.

TROCAR

As guias ou novas linhas são substituídas pelo número apropriado de espaços.

PRESERVE

Retém qualquer espaço em branco no valor dos dados.

Se o parâmetro **CHAR-WHITESPACE** não for configurado, o espaço em branco será reduzido.

Nota: Esse parâmetro não se aplica aos campos com um formato de date-time, uri, base64Binary ou hexBinary, em que o espaço em branco é sempre reduzido.

CONTID = *value*

Em um provedor de serviços, especifica o nome do contêiner que contém a estrutura de dados de nível superior que é usada para representar uma mensagem JSON.

O comprimento do contêiner que o CICS transmite para o programa de aplicativo de destino é o maior dos comprimentos do contêiner de solicitação e do contêiner de resposta.

DATA-SCREENING = { **ENABLED** | **DISABLED** }

Especifica se os dados fornecidos pelo aplicativo foram verificados com relação aos erros.

ENABLED

Qualquer dado de tempo de execução inconsistente com a estrutura de linguagem é tratado como um erro e a mensagem DFHPI1010 é emitida. Uma resposta de erro é retornada ao aplicativo.

DISABLED

Os valores nos dados de tempo de execução fornecidos pelo aplicativo que são inconsistentes com a estrutura de linguagem são substituídos pelos valores padrão. Por exemplo, um zero substitui um valor inválido em um campo numérico. A mensagem DFHPI1010 não é emitida e uma resposta normal é retornada ao aplicativo. Esse recurso pode ser usado para evitar as respostas de erro INVALID_PACKED_DEC e INVALID_ZONED_DEC que são geradas a partir de campos de saída não inicializados.

DATA-TRUNCATION = { **DISABLED** | **ENABLED** }

Especifica se dados de comprimento variável são tolerados em uma estrutura de campo de comprimento fixo:

DISABLED

Se os dados forem menores do que o comprimento fixado esperado pelo CICS, o CICS rejeitará os dados truncados e emitirá uma mensagem de erro.

ENABLED

Se os dados são menores do que o comprimento fixo que o CICS está esperando, o CICS tolera os dados truncados e processa os dados ausentes como valores nulos.

DATETIME = { **PACKED15** | **STRING** }

Especifica como elementos de data/hora JSON são mapeados para a estrutura de linguagem.

PACKED15

O padrão é que qualquer elemento de data/hora JSON é processado como um registro de data e hora e é mapeado para o formato ABSTIME do CICS.

CADEIA

O elemento de data/hora JSON é processado como texto.

DEFAULT-ARRAY-MAXITEMS = *value*

Especifica o limite máximo da matriz a ser aplicado quando nenhuma informação de ocorrência máxima (maxItems) estiver implícita no esquema JSON. Se esse parâmetro não estiver configurado, nenhum limite máximo será aplicado. O valor desse parâmetro pode ser um número inteiro positivo no intervalo de 1 a 2147483647. Esse parâmetro pode ser combinado com o uso do parâmetro **INLINE-MAXOCCURS-LIMIT** para influenciar como as matrizes JSON são mapeadas para as estruturas de linguagem.

DEFAULT-CHAR-MAXLENGTH = { **255** | *value* }

Especifica o comprimento da matriz padrão de dados de caractere em caracteres para mapeamentos em que nenhum comprimento esteja implícito no documento de descrição de serviço da web, quando o nível de mapeamento é 1.2 ou superior. O valor desse parâmetro pode ser um número inteiro positivo no intervalo de 1 a 2.147.483.647.

DEFAULT-FRACTION-DIGITS = { **3** | *value* }

Especifica o número padrão de dígitos fracionários para uso em um tipo de esquema decimal XML. O padrão é 3. Para COBOL, o intervalo válido é de 0 a 17 ou de 0 a 30 se o parâmetro **WIDE-COMP3** está sendo usado. Para C ou PLI, o intervalo válido é de 0 a 30.

INLINE-MAXOCCURS-LIMIT = { **1** | *value* }

Especifica se o conteúdo de repetição de variável sequencial é usado com base na palavra-chave do esquema JSON maxItems. O conteúdo de repetição variável que é mapeado sequencialmente é colocado no contêiner atual com o resto da estrutura de linguagem gerada. O conteúdo de repetição variável é armazenado em duas partes, como um contador que armazena o número de ocorrências dos dados e como uma matriz que armazena cada ocorrência dos dados. A alternativa de mapeamento para conteúdo variavelmente repetido é o mapeamento baseado em contêiner, que armazena o número de ocorrências dos dados e o nome do contêiner onde os dados são colocados. Armazenar os dados em um contêiner separado tem implicações de desempenho que podem tornar o mapeamento sequencial preferível.

O parâmetro **INLINE-MAXOCCURS-LIMIT** está disponível somente no nível de mapeamento 2.1 em diante. O valor de **INLINE-MAXOCCURS-LIMIT** pode ser um número inteiro positivo no intervalo de 0 a 32.767. Um valor 0 indica que o mapeamento sequencial não é usado. O valor de 1 assegura que elementos opcionais sejam mapeados de forma sequencial. Se o *value* do atributo maxOccurs for maior do que o *value* de **INLINE-MAXOCCURS-LIMIT**, o mapeamento baseado em contêiner será usado; caso contrário, o mapeamento sequencial será usado.

Ao decidir se você deseja que listas variavelmente repetidas sejam mapeadas sequencialmente, considere o comprimento de um único item de dados recorrentes. Se poucas instâncias de comprimento longo ocorrerem, o mapeamento baseado em contêiner é preferível; se muitas instâncias de comprimento curto ocorrerem, o mapeamento sequencial é preferível.

JSON-SCHEMA-REQUEST = *value*

Este é um parâmetro obrigatório.

O valor indica o local do UNIX System Services no qual o esquema JSON da solicitação é armazenado.

JSON-SCHEMA-RESPONSE = *value*

Este é um parâmetro obrigatório.

O valor indica o local do UNIX System Services no qual o esquema JSON de resposta é armazenado.

LANG = **COBOL**

Especifica que a linguagem de programação da estrutura de linguagem de alto nível é COBOL.

LANG = **PLI-ENTERPRISE**

Especifica que a linguagem de programação da estrutura de linguagem de alto nível é Enterprise PL/I.

LANG = **PLI-OTHER**

Especifica que a linguagem de programação da estrutura de linguagem de alto nível é um nível de PL/I diferente de Enterprise PL/I.

LANG = **C**

Especifica que a linguagem de programação da estrutura de linguagem de alto nível é C.

LANG = **CPP**

Especifica que a linguagem de programação da estrutura de linguagem de alto nível é C++.

LOGFILE = *value*

O nome completo do z/OS UNIX do arquivo no qual DFHJS2LS grava seu log de atividades e suas informações de rastreamento. DFHJS2LS cria o arquivo, mas não a estrutura de diretório, se ela não existir.

Normalmente, você não usa esse arquivo, mas ele poderá ser solicitado pela organização de serviço IBM caso você encontre problemas com o DFHJS2LS.

MAPPING-LEVEL = { **1.0** | **1.1** | **1.2** | **2.0** | **2.1** | **2.2** | **3.0** | **4.0** | **4.1** }

Especifica o nível de mapeamento que DFHJS2LS usa ao gerar o arquivo de ligação de serviço da web e a estrutura de linguagem. Você pode selecionar estas opções:

1.0 Essa opção é mantida para compatibilidade com serviços da web SOAP. Ela não é recomendada para uso com JSON.

1.1 Essa opção é mantida para compatibilidade com serviços da web SOAP. Ela não é recomendada para uso com JSON.

1.2 Essa opção é mantida para compatibilidade com serviços da web SOAP. Ela não é recomendada para uso com JSON.

2.0 Essa opção é mantida para compatibilidade com serviços da web SOAP. Ela não é recomendada para uso com JSON.

- 2.1 Essa opção é mantida para compatibilidade com serviços da web SOAP. Ela não é recomendada para uso com JSON.
- 2.2 Essa opção é mantida para compatibilidade com serviços da web SOAP. Ela não é recomendada para uso com JSON.
- 3,0 Use este nível de mapeamento para gerar o esquema JSON usando o conjunto completo de opções disponíveis.
- 4.0 Use este nível de mapeamento com uma região do CICS TS 5.2 quando você desejar usar UTF-16.
- 4.1 Para suporte de matriz truncável, use este nível de mapeamento com uma região CICS V5.2 que tenha o APAR PI67641 aplicado ou qualquer região CICS V5.3 ou mais recente.

Para obter mais informações sobre níveis de mapeamento, consulte Níveis de mapeamento para os assistentes CICS.

MAPPING-OVERRIDES = { SAME-AS-MAPPING-LEVEL | HYPHENS-AS-UNDERSCORES | INTEGER-AS-PIC9 | LESS-DUP-NAMES | UNDERSCORES-AS-HYPHENS }

Especifica se o comportamento padrão é substituído para o nível de mapeamento especificado ao gerar estruturas de linguagem.

SAME-AS-MAPPING-LEVEL

Este parâmetro gera estruturas de linguagem no mesmo estilo que o nível de mapeamento. Este é o padrão.

HYPHENS-AS-UNDERSCORES

Para PL/I somente. Este parâmetro converte quaisquer hifens no esquema JSON em sublinhados em vez do caractere X, para melhorar a capacidade de leitura das estruturas de linguagem PL/I geradas. Para obter mais informações, consulte Esquema JSON para mapeamento de PL/I.

INTEGER-AS-PIC9

Apenas para COBOL e DFHJS2LS. Este parâmetro gera estruturas de linguagem que contêm valores de número inteiro do esquema JSON como numerais em vez de caracteres alfanuméricos.

LESS-DUP-NAMES

Este parâmetro gera nomes de campo de estrutura não estruturais com `_value` no final do nome para ativar a referência direta ao campo. Por exemplo, na estrutura de linguagem PLI a seguir, quando `MAPPING-OVERRIDES=LESS-DUP-NAMES` é especificado, o campo de nível 12 `streetName` é sufixado com `_value`:

```
09 streetName,
12 streetName CHAR(255) VARYING
UNALIGNED,
12 filler BIT (7),
12 attr_nil_streetName_value BIT (1),
```

A estrutura resultante é conforme a seguir:

```
09 streetName,
12 streetName_value CHAR(255) VARYING
UNALIGNED,
12 filler BIT (7),
12 attr_nil_streetName_value BIT (1),
```

UNDERSCORES-AS-HYPHENS

Esta opção é ativada automaticamente no nível de mapeamento 4.0.

Apenas para COBOL. Este parâmetro converte quaisquer sublinhados no esquema JSON em hifens, em vez do caractere X, para melhorar a capacidade de leitura das estruturas de linguagem COBOL geradas. Se ocorrerem quaisquer conflitos de nomes de campo, os campos serão numerados para garantir que sejam exclusivos. Para obter mais informações, consulte Esquema JSON para mapeamento COBOL.

MINIMUM-RUNTIME-LEVEL = { MINIMUM | 1.0 | 1.1 | 1.2 | 2.0 | 2.1 | 2.2 | 3.0 | 4.0 | 4.1 | **CURRENT** }

Especifica o ambiente de tempo de execução mínimo do CICS no qual o arquivo de ligação de serviço da web pode ser implementado. Se você selecionar um nível que não corresponde aos outros parâmetros que você especificou, receberá uma mensagem de erro. É possível selecionar essas opções:

MINIMUM

O menor nível de tempo de execução do CICS possível é alocado automaticamente conforme os parâmetros que você selecionou.

1.0 Essa opção é mantida para compatibilidade com serviços da web SOAP. Ela não é recomendada para uso com JSON.

1.1 Essa opção é mantida para compatibilidade com serviços da web SOAP. Ela não é recomendada para uso com JSON.

1.2 Essa opção é mantida para compatibilidade com serviços da web SOAP. Ela não é recomendada para uso com JSON.

2.0 Essa opção é mantida para compatibilidade com serviços da web SOAP. Ela não é recomendada para uso com JSON.

2.1 Essa opção é mantida para compatibilidade com serviços da web SOAP. Ela não é recomendada para uso com JSON.

2.2 Essa opção é mantida para compatibilidade com serviços da web SOAP. Ela não é recomendada para uso com JSON.

3.0 O arquivo de ligação de serviço da web gerado é implementado em uma região do CICS TS 4.1 ou mais recente.

Nota: O suporte JSON está disponível somente a partir do CICS TS 4.2 em diante.

4.0 O arquivo de ligação de serviço da web gerado é implementado com sucesso em um CICS TS 5.2 ou em uma região posterior. Com este nível de tempo de execução, é possível usar um nível de mapeamento 4.0 ou anterior para o parâmetro **MAPPING-LEVEL**. É possível utilizar qualquer parâmetro opcional neste nível.

4.1 O arquivo de ligação de serviço da web gerado é implementado com sucesso em uma região CICS TS V5.2 que tem o APAR PI67641 aplicado ou em qualquer região CICS TS V5.3 ou mais recente. Com este nível de tempo de execução, é possível utilizar um nível de mapeamento 4.1 ou anterior para o parâmetro **MAPPING-LEVEL**.

CURRENT

O arquivo de ligação de serviço da web gerado é implementado com sucesso em uma região CICS no mesmo nível de tempo de execução que você está usando para gerar o arquivo de ligação de serviço da web.

NAME-TRUNCATION = { **LEFT** | **RIGHT** }

Especifica se os nomes JSON são truncados a partir da esquerda ou da direita. O assistente de serviço da web do CICS trunca nomes JSON até o comprimento apropriado para a linguagem de alto nível especificada; por padrão, os nomes são truncados a partir da direita.

PDSCP = *value*

Especifica a página de códigos que é usada nos membros do conjunto de dados particionados que são especificados nos parâmetros **REQMEM** e **RESPMEM**, em que *value* é um número de CCSID ou um número da página de códigos Java. Se este parâmetro não for especificado, a página de códigos do z/OS UNIX System Services será usada. Por exemplo, você pode especificar **PDSCP** = 037.

PDSLIB = *value*

Especifica o nome do conjunto de dados particionados que contém a linguagem de alto nível gerada. Os membros do conjunto de dados que são usados para a solicitação e a resposta são especificados nos parâmetros **REQMEM** e **RESPMEM**.

PGMINT = { **CHANNEL** | **COMMAREA** }

Para um provedor de serviços, especifica como o CICS transmite dados ao programa de aplicativo de destino:

CHANNEL

CICS usa uma interface do canal para transmitir dados ao programa de aplicativo de destino.

COMMAREA

CICS usa uma área de comunicação para transmitir dados ao programa de aplicativo de destino.

Quando o programa de aplicativo de destino tiver processado a solicitação, ele deverá usar o mesmo mecanismo para retornar a resposta. Se a solicitação foi recebida em uma área de comunicação, a resposta deverá ser retornada na área de comunicação; se a solicitação foi recebida em um contêiner, a resposta deverá ser retornada em um contêiner. O comprimento da área de comunicação ou contêiner que o CICS transmite para o programa de aplicativo de destino é o maior dos comprimentos da área ou do contêiner de comunicação da solicitação e da área ou contêiner de comunicação de resposta.

PGMNAME = *value*

Especifica o nome de um recurso do PROGRAMA CICS.

Quando DFHJS2LS é usado para gerar um arquivo de ligação de serviço da web que é usado em um provedor de serviços, você deve fornecer esse parâmetro. Ele especifica o nome do recurso do programa de aplicativo que é exposto como um serviço da web.

Quando DFHJS2LS for usado para gerar um arquivo de ligação de serviço da web que é usado em um solicitante de serviço, omita este parâmetro.

REQMEM = *value*

Especifica um prefixo de um a seis caracteres que o DFHJS2LS usa para gerar os nomes dos membros do conjunto de dados particionados que contém as estruturas de linguagem de alto nível para a solicitação de serviço da web, que são os dados de entrada para o programa de aplicativo.

O DFHJS2LS gera o nome do membro do conjunto de dados particionados anexando 01 ao prefixo.

RESPMEM = *value*

Especifica um prefixo de um a seis caracteres que o DFHJS2LS usa para gerar os nomes dos membros do conjunto de dados particionados que contém as estruturas de linguagem de alto nível para a resposta de serviço da web, que são os dados de saída do programa de aplicativo.

O DFHJS2LS gera o nome do membro do conjunto de dados particionados anexando 01 ao prefixo.

STRUCTURE = (*request* , *response*)

Apenas para C e C++, especifica como os nomes das estruturas de solicitação e resposta são gerados.

as estruturas de solicitação e resposta geradas recebem os nomes *request01* e *response01*.

Se um ou ambos os nomes forem omitidos, as estruturas terão o mesmo nome que os nomes do membro do conjunto de dados particionados gerados a partir dos parâmetros **REQMEM** e **RESPMEM** especificados por você.

SYNCONRETURN = { **NO** | **YES** }

Especifica se o serviço da web remoto pode emitir um ponto de sincronização.

NO O serviço da web remoto não pode emitir um ponto de sincronização. Este valor é o padrão. Se o serviço da web remoto emitir um ponto de sincronização, ele falhará com um encerramento anormal de ADPL.

YES O serviço da web remoto pode emitir um ponto de sincronização. Se você selecionar YES, a tarefa remota é confirmada como uma unidade de trabalho separada quando o controle retorna do serviço da web remoto. Se o serviço da web remoto atualizar um recurso recuperável e ocorrer uma falha após ele retornar, a atualização para esse recurso não poderá ser restaurada.

TRANSACTION = *name*

Em um provedor de serviços, este parâmetro especifica um nome com um a quatro caracteres de uma transação de alias que pode iniciar o pipeline. O valor desse parâmetro é usado para definir o atributo TRANSACTION do recurso URIMAP quando ele é criado automaticamente usando o comando de varredura **PIPELINE**.

Caracteres aceitáveis:

A-Z a-z 0-9 \$ @ # _ < >

URI = *value*

Em um provedor de serviços, este parâmetro especifica o URI relativo que um cliente usa para acessar o serviço da web. O CICS usa o valor que é especificado quando ele gera um recurso URIMAP a partir do arquivo de ligação de serviço da web que é criado pelo DFHJS2LS. O parâmetro especifica o componente de caminho do URI ao qual a definição de URIMAP se aplica.

USERID = *id*

Em um provedor de serviços, este parâmetro especifica um ID do usuário de um a oito caracteres, o qual pode ser usado por qualquer Web client. Para uma resposta gerada por aplicativo ou um serviço da web, a transação de alias é conectada com este ID do usuário. O valor desse parâmetro é usado para definir o atributo USERID do recurso URIMAP quando ele é criado automaticamente usando o comando de varredura **PIPELINE**.

Caracteres aceitáveis:

A-Z a-z 0-9 \$ @ #

WIDE-COMP3 = { **FULL** | **NO** | **YES** }

Controla o tamanho máximo do comprimento variável decimal compactado na estrutura de linguagem COBOL ou PL/I gerada.

FULL Para COBOL e PL/I. DFHJS2LS gera um campo decimal compactado que é grande o suficiente para conter todos os valores válidos. O tamanho máximo são 31 dígitos. Este é o padrão.

NO Apenas para COBOL. DFHJS2LS limita o comprimento variável decimal compactado a 18 ao gerar a estrutura de linguagem COBOL tipo COMP-3. Se o tamanho decimal compactado for maior que 18, a mensagem DFHPI9022W será emitida para indicar que o tipo especificado está sendo limitado a um total de 18 dígitos.

YES Apenas para COBOL. DFHJS2LS suporta o tamanho máximo de 31 ao gerar a estrutura de linguagem COBOL tipo COMP-3.

Nota: As opções NO e YES geram campos que são incapazes de representar todos os valores válidos; a opção FULL evita esse problema. No entanto, a opção FULL permite que alguns valores inválidos sejam representados no campo decimal compactado. Por exemplo, se um esquema indica que há um máximo de cinco dígitos e um máximo de dois dígitos fracionários, a opção FULL gerará um campo decimal compactado que permite sete dígitos e isso permite espaço para valores válidos como 25000 e 999,99, mas também fornece espaço para alguns valores inválidos como 9999,99. Quando você usar a opção FULL, tome cuidado para não gerar valores inválidos em dados do aplicativo.

WSBIND = *value*

O nome completo do z/OS UNIX do arquivo de ligação de serviço da web. DFHJS2LS cria o arquivo, mas não a estrutura de diretório, se ela não existir. A extensão do arquivo é padronizada como .wsbind.

Outras informações

- O ID do usuário sob o qual DFHJS2LS é executado deve ser configurado para usar o UNIX System Services. O ID do usuário deve ter permissão de leitura para a estrutura de arquivo e bibliotecas PDS do CICS z/OS UNIX e permissão de gravação para os diretórios especificados nos parâmetros **LOGFILE**, **WSBIND** e **WSDL**.
- O ID do usuário deve ter uma alocação de armazenamento suficientemente grande para executar Java.
- A JCL possui um comprimento de parâmetro máximo de 100 caracteres. Isso pode ser aumentado usando a instrução **STDPARM**. Para obter mais informações, consulte *z/OS UNIX System Services User Guide*.

Exemplo

```
//JS2LS JOB '  
accounting information  
,  
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: Esquema JSON para conversão de linguagem de alto nível para serviços RESTful

O procedimento DFHJS2LS gera uma estrutura de dados de linguagem de alto nível e um arquivo de ligação de serviço da web a partir de um esquema JSON. É possível utilizar DFHJS2LS ao se preparar para criar um provedor de serviços RESTful JSON. Este tópico lista as instruções de controle de tarefa, parâmetros simbólicos, parâmetros de entrada e suas descrições para DFHJS2LS.

O procedimento JCL DFHJS2LS é instalado no conjunto de dados *HLQ.XDFHINST*, em que *HLQ* é o qualificador de alto nível no qual o CICS está instalado.

Instruções de controle de tarefa para DFHJS2LS

JOB Inicia a tarefa.

EXEC Especifica o nome do procedimento (DFHJS2LS).

INPUT.SYSUT1 DD

Especifica a entrada. Os parâmetros de entrada geralmente são especificados no fluxo de entrada. No entanto, eles podem ser definidos em um conjunto de dados ou em um membro de um conjunto de dados particionados.

Parâmetros simbólicos

Os parâmetros simbólicos a seguir são definidos no DFHJS2LS:

JAVADIR = *path*

Especifica o nome do diretório Java que é usado pelo DFHJS2LS. O valor desse parâmetro é anexado a */usr/lpp/* para produzir um nome de caminho completo de */usr/lpp/ path*.

Normalmente, você não especifica este parâmetro. O valor padrão é o valor que foi fornecido à tarefa de instalação do CICS (DFHISTAR) no parâmetro **JAVADIR**.

PATHPREFIX = *prefix*

Especifica um prefixo que estende o caminho do diretório z/OS UNIX que é usado em outros parâmetros ou '' (sequência de caracteres vazia) se nenhum prefixo é usado.

Normalmente, você não especifica este parâmetro. O valor padrão é o valor que foi fornecido para a tarefa de instalação do CICS (DFHISTAR) no parâmetro **PATHPREFIX**.

SERVICE = *value*

Use este parâmetro somente quando orientado a fazer isso pelo Suporte IBM.

TMPDIR = *tmpdir*

Especifica o local de um diretório no z/OS UNIX que o DFHJS2LS utiliza como uma área de trabalho provisória. O ID do usuário sob o qual a tarefa é executada deve ter permissão de leitura e gravação para esse diretório.

O valor padrão é /tmp.

TMPPFILE = *tmpprefix*

Especifica um prefixo que o DFHJS2LS usa para construir os nomes dos arquivos de área de trabalho provisória.

O valor padrão é JS2LS.

USSDIR = *path*

Especifica o nome do diretório TS do CICS no sistema de arquivos do UNIX System Services. O valor desse parâmetro é anexado a /usr/lpp/cicsts/ para produzir um nome do caminho completo de /usr/lpp/cicsts/ *path* . Isso deve ser especificado como ' .' (ponto) se o padrão for usado.

Normalmente, você não especifica este parâmetro. O valor padrão é o valor que foi fornecido à tarefa de instalação do CICS (DFHISTAR) no parâmetro **USSDIR**.

O Espaço de Trabalho Temporário

DFHJS2LS cria os três arquivos temporários a seguir no tempo de execução:

```
tmpdir / tmpprefix .in  
tmpdir / tmpprefix .out  
tmpdir / tmpprefix .err
```

em que:

tmpdir é o valor especificado no parâmetro **TMPDIR**.

tmpprefix é o valor especificado no parâmetro **TMPPFILE**.

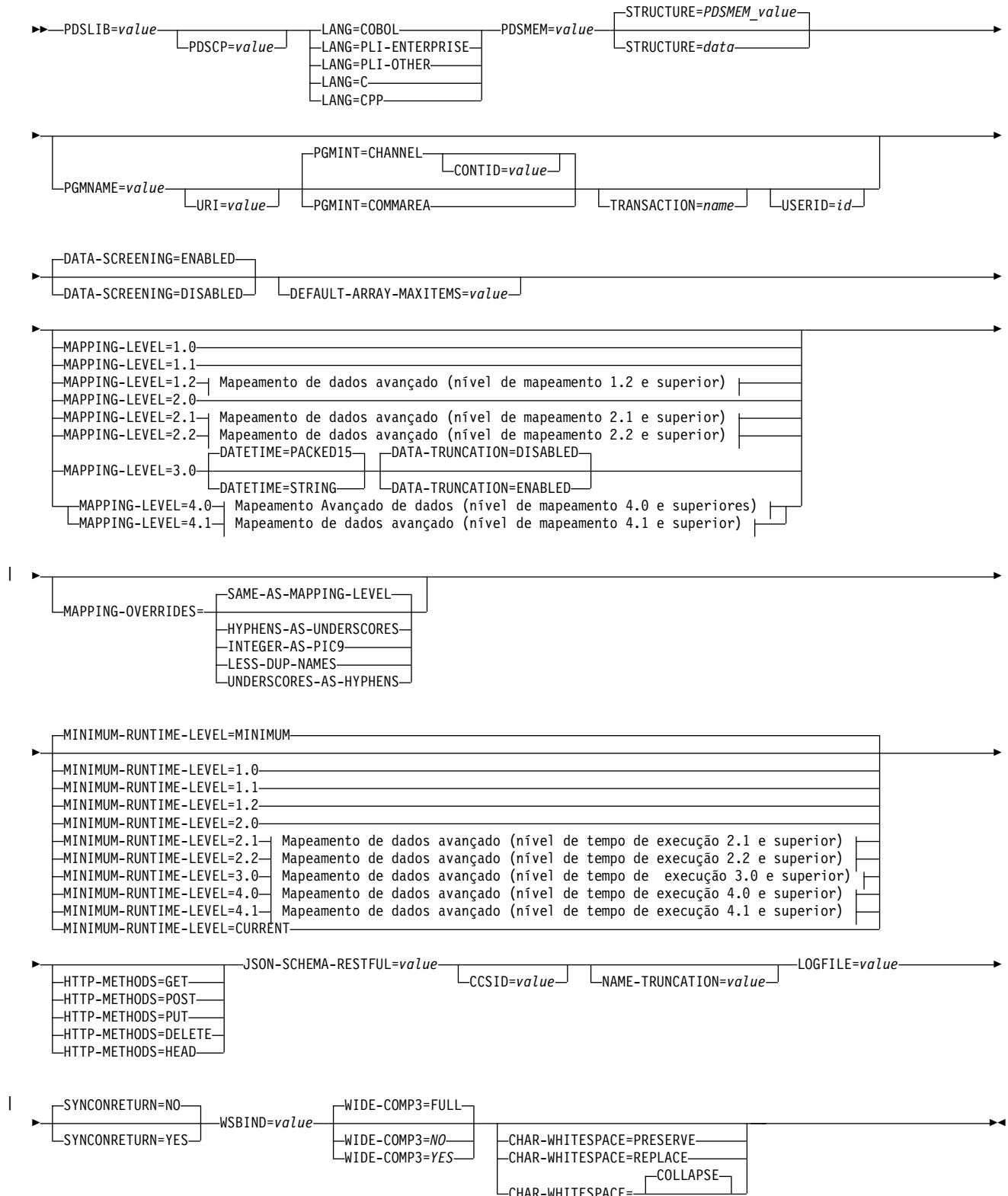
Os nomes padrão para os arquivos quando **TMPDIR** e **TMPPFILE** não são especificados, são conforme a seguir:

```
/tmp/JS2LS.in  
/tmp/JS2LS.out  
/tmp/JS2LS.err
```

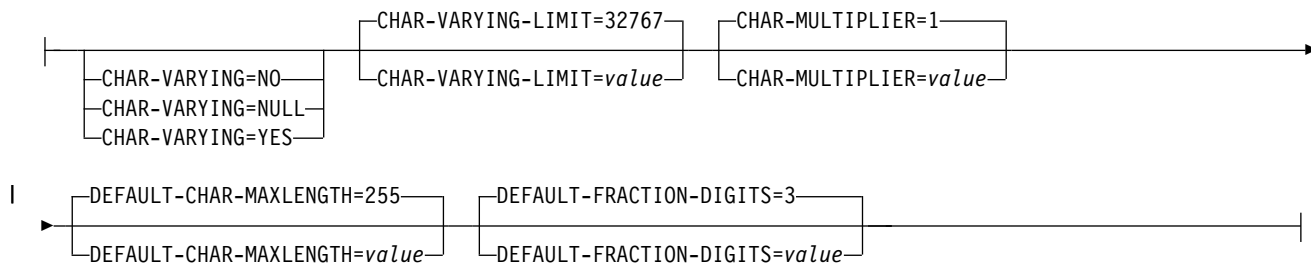
Importante: O DFHJS2LS não bloqueia o acesso aos arquivos do z/OS UNIX ou aos membros do conjunto de dados. Portanto, se duas ou mais instâncias de DFHJS2LS são executadas simultaneamente e usam os mesmos arquivos da área de trabalho provisória, nada impede que uma tarefa sobrescreva os arquivos da área de trabalho enquanto outra tarefa está usando-os, levando a falhas imprevisíveis.

Portanto, você será avisado para planejar uma convenção de nomenclatura e procedimentos operacionais que evitam essa situação. Por exemplo, é possível usar o parâmetro simbólico do sistema **SYSUID** para gerar nomes de arquivos de área de trabalho que são exclusivos para um usuário individual. Estes arquivos temporários são excluídos antes do encerramento da tarefa.

Parâmetros de entrada para DFHJS2LS



Mapeamento de Dados Avançado (Nível de Mapeamento 1.2 e Superior):



Mapeamento de Dados Avançado (Nível de Mapeamento 2.1 e Superior):



Utilização de Parâmetros

- Você pode especificar os parâmetros de entrada em qualquer ordem.
- Cada parâmetro deve ser iniciado em uma nova linha.
- Um parâmetro, e seu caractere de continuação, se você usar um, não deve estender além da coluna 72; as colunas 73 a 80 devem conter espaços em branco.
- Se um parâmetro é muito longo para se ajustar em uma única linha, use um caractere de asterisco (*) no final da linha para indicar que o parâmetro continua na próxima linha. Tudo, incluindo espaços antes do asterisco, é considerado parte do parâmetro. Por exemplo:

```
WSBIND=wsbinddir*  
      /app1
```

é equivalente a

```
WSBIND=wsbinddir/app1
```

- Um caractere # na primeira posição de caractere da linha é um caractere de comentário. A linha é ignorada.
- Uma vírgula na posição de último caractere da linha é um separador de linha opcional e é ignorado.

Descrições de Parâmetros

CCSID = *value*

Especifica o CCSID que é usado no tempo de execução para codificar dados de caractere na estrutura de dados do aplicativo. O valor desse parâmetro substitui o valor do parâmetro de inicialização do sistema **LOCALCCSID**. O *value* deve ser um CCSID EBCDIC suportado por Java e . Se você não especificar esse parâmetro, a estrutura de dados do aplicativo será codificada usando o CCSID especificado no parâmetro de inicialização do sistema.

CHAR-MULTIPLIER = { 1 | *value* }

Especifica o número de bytes a ser permitido para cada caractere quando o nível de mapeamento é 1.2 ou mais recente. O *value* deste parâmetro pode ser um número inteiro positivo no intervalo de 1 a 2.147.483.647. Todos os mapeamentos baseados em caracteres não numéricos estão sujeitos a este multiplicador. Campos binários, numéricos, zoneados e decimais compactados não estão sujeitos a este multiplicador.

Este parâmetro pode ser útil se, por exemplo, você está planejando usar caracteres DBCS onde pode optar por um multiplicador de 3 para permitir espaço para possíveis caracteres shift-out e shift-in em cada caractere de byte duplo no tempo de execução.

Quando você configura **CCSID=1200** (indicando UTF-16), os únicos valores válidos para **CHAR-MULTIPLIER** são 2 ou 4. Ao usar UTF-16, o valor padrão é 2. Use **CHAR-MULTIPLIER=2** quando esperar que os dados do aplicativo contenham caracteres que requerem 1 unidade de codificação UTF-16. Use **CHAR-MULTIPLIER=4** quando esperar que os dados do aplicativo contenham caracteres que requerem 2 unidades de codificação UTF-16.

Nota: A configuração de **CHAR-MULTIPLIER** como 1 não impede o uso de caracteres DBCS e sua configuração como 2 não impede o uso de pares substitutos UTF-16. No entanto, se caracteres largos forem usados rotineiramente, alguns valores válidos não se ajustarão no campo alocado. Se um valor **CHAR-MULTIPLIER** maior for usado, poderá ser possível armazenar mais caracteres no campo alocado que são válidos no XML. Deve-se tomar cuidado com relação à conformidade com as restrições de intervalo apropriadas.

CHAR-VARYING = NO | NULL | YES

Especifica como os dados de caracteres de comprimento variável são mapeados quando o nível de mapeamento é 1.2 ou superior. Os tipos de dados binários de comprimento variável são sempre mapeados para um contêiner ou uma estrutura variável. Se você não especificar esse parâmetro, o mapeamento padrão dependerá da linguagem especificada. Você pode selecionar estas opções:

NO Os dados de caractere de comprimento variável são mapeados como sequências de comprimento fixo.

NULL Os dados de caractere de comprimento variável são mapeados para sequências com terminação nula.

YES Dados de caractere de comprimento variável são mapeados para um tipo de dados CHAR VARYING em PL/I. Nas linguagens COBOL, C e C++, os dados de caracteres de comprimento variável são mapeados para uma representação equivalente que consiste em dois elementos relacionados: dados de comprimento e dados.

CHAR-VARYING-LIMIT = 32767 | value

Especifica o tamanho máximo de dados binários e de dados de caractere de comprimento variável que são mapeados para a estrutura de linguagem quando o nível de mapeamento é 1.2 ou superior. Se os dados de caractere ou binários forem maiores do que o valor especificado neste parâmetro, eles serão mapeados para um contêiner e o nome do contêiner será usado na estrutura de linguagem gerada. O valor pode variar de 0 até o padrão de 32.767 bytes.

CHAR-WHITESPACE = COLLAPSE | REPLACE | PRESERVE

Especifica como espaços em branco em valores de sequência de tipos serão manipulados pelo CICS.

COLLAPSE

Espaços em branco iniciais, finais e integrados são removidos e todas as guias, novas linhas e espaços consecutivos são substituídos por caracteres de espaço único.

TROCAR

As guias ou novas linhas são substituídas pelo número apropriado de espaços.

PRESERVE

Retém qualquer espaço em branco no valor dos dados.

Se o parâmetro **CHAR-WHITESPACE** não for configurado, o espaço em branco será reduzido.

Nota: Este parâmetro não se aplica aos campos com um formato de data/hora , URI , base64Binary ou hexBinary , em que o espaço em branco é sempre reduzido.

CONTID = *value*

Em um provedor de serviços, especifica o nome do contêiner que contém a estrutura de dados de nível superior que é usada para representar uma mensagem JSON.

O comprimento do contêiner que o CICS transmite para o programa de aplicativo de destino é o maior dos comprimentos do contêiner de solicitação e do contêiner de resposta.

DATA-SCREENING = { ENABLED | DISABLED }

Especifica se os dados fornecidos pelo aplicativo foram verificados com relação aos erros.

ENABLED

Quaisquer dados inconsistentes de tempo de execução fornecidos pelo aplicativo com a estrutura de linguagem são tratados como um erro e a mensagem DFHPI1010 é emitida. Uma resposta de erro é retornada ao aplicativo.

DISABLED

Os valores dos dados de tempo de execução fornecidos pelo aplicativo que são inconsistentes com a estrutura de linguagem são substituídos pelos valores padrão. Por exemplo, um zero substitui um valor inválido em um campo numérico. A mensagem DFHPI1010 não é emitida e uma resposta normal é retornada ao aplicativo. Esse recurso pode ser usado para evitar as respostas de erro INVALID_PACKED_DEC e INVALID_ZONED_DEC que são geradas a partir de campos de saída não inicializados.

DATA-TRUNCATION = { DISABLED | ENABLED }

Especifica se dados de comprimento variável são tolerados em uma estrutura de campo de comprimento fixo:

DISABLED

Se os dados forem menores do que o comprimento fixo que o CICS está esperando, o CICS rejeitará os dados truncados e emitirá uma mensagem de erro.

ENABLED

Se os dados são menores do que o comprimento fixo que o CICS está esperando, o CICS tolera os dados truncados e processa os dados ausentes como valores nulos.

DATETIME = PACKED15 | STRING

Especifica como elementos de data/hora JSON são mapeados para a estrutura de linguagem.

PACKED15

O padrão é que qualquer elemento de data/hora JSON é processado como um registro de data e hora e é mapeado para o formato ABSTIME do CICS.

CADEIA

O elemento de data/hora JSON é processado como texto.

DEFAULT-ARRAY-MAXITEMS = value

Especifica o limite máximo da matriz a ser aplicado quando nenhuma informação de ocorrência máxima (maxItems) estiver implícita no esquema JSON. Se esse parâmetro não estiver configurado, nenhum limite máximo será aplicado. O valor desse parâmetro pode ser um número inteiro positivo no intervalo de 1 a 2147483647. Esse parâmetro pode ser combinado com o uso do parâmetro **INLINE-MAXOCCURS-LIMIT** para influenciar como as matrizes JSON são mapeadas para as estruturas de linguagem.

DEFAULT-CHAR-MAXLENGTH = 255 | value

Especifica o comprimento da matriz padrão de dados de caractere em caracteres para mapeamentos em que nenhum comprimento esteja implícito no documento de descrição de serviço da web, quando o nível de mapeamento é 1.2 ou superior. O valor desse parâmetro pode ser um número inteiro positivo no intervalo de 1 a 2.147.483.647.

DEFAULT-FRACTION-DIGITS = { 3 | value }

Especifica o número padrão de dígitos fracionários para uso em um tipo de esquema decimal XML. O padrão é 3. Para COBOL, o intervalo válido é de 0 a 17 ou de 0 a 30 se o parâmetro **WIDE-COMP3** está sendo usado. Para C ou PLI, o intervalo válido é de 0 a 30.

HTTP-METHODS = { GET | POST | PUT | DELETE | HEAD }, { GET | POST | PUT | DELETE | HEAD }, *

Esse é um parâmetro opcional.

O valor padrão é para GET, POST, PUT e DELETE serem configurados, o que informa ao DFHJS2LS que o aplicativo suporta as quatro operações RESTful principais.

Se um valor for fornecido, DFHJS2LS constrói um arquivo WSBind no qual somente os métodos HTTP especificados explicitamente são aceitos.

Se um aplicativo desejar implementar o método HEAD, ele deverá optar deliberadamente por fazer isso. Por padrão, o DFHJS2LS assume que o aplicativo não suporta métodos HTTP HEAD recebidos.

Se um cliente JSON tentar usar um método HTTP não suportado, o CICS responderá com uma resposta HTTP 405.

INLINE-MAXOCCURS-LIMIT = 1 | value

Especifica se o conteúdo de repetição de variável sequencial é usado com base na palavra-chave do esquema JSON maxItems. O conteúdo de repetição variável que é mapeado sequencialmente é colocado no contêiner atual com o resto da estrutura de linguagem gerada. O conteúdo de repetição variável é armazenado em duas partes, como um contador que armazena o número de ocorrências dos dados e como uma matriz que armazena cada ocorrência dos dados. A alternativa de mapeamento para conteúdo variavelmente repetido é o mapeamento baseado em contêiner, que armazena o número de ocorrências dos dados e o nome do contêiner onde os dados são colocados. Armazenar os dados em um contêiner separado tem implicações de desempenho que podem tornar o mapeamento sequencial preferível.

O parâmetro **INLINE-MAXOCCURS-LIMIT** está disponível somente no nível de mapeamento 2.1 em diante. O valor de **INLINE-MAXOCCURS-LIMIT** pode ser um

número inteiro positivo no intervalo de 0 a 32.767. Um valor 0 indica que o mapeamento sequencial não é usado. O valor de 1 assegura que elementos opcionais sejam mapeados de forma sequencial. Se o *value* do atributo `maxOccurs` for maior do que o *value* de **INLINE-MAXOCCURS-LIMIT**, o mapeamento baseado em contêiner será usado; caso contrário, o mapeamento sequencial será usado.

Ao decidir se você deseja que listas variavelmente repetidas sejam mapeadas sequencialmente, considere o comprimento de um único item de dados recorrentes. Se poucas instâncias de comprimento longo ocorrerem, o mapeamento baseado em contêiner é preferível; se muitas instâncias de comprimento curto ocorrerem, o mapeamento sequencial é preferível.

JSON-SCHEMA-RESTFUL = *value*

Este é um parâmetro obrigatório.

O valor indica o local do UNIX System Services no qual o esquema JSON de resposta é armazenado.

LANG = **COBOL**

Especifica que a linguagem de programação da estrutura de linguagem de alto nível é COBOL.

LANG = **PLI-ENTERPRISE**

Especifica que a linguagem de programação da estrutura de linguagem de alto nível é Enterprise PL/I.

LANG = **PLI-OTHER**

Especifica que a linguagem de programação da estrutura de linguagem de alto nível é um nível de PL/I diferente de Enterprise PL/I.

LANG = **C**

Especifica que a linguagem de programação da estrutura de linguagem de alto nível é C.

LANG = **CPP**

Especifica que a linguagem de programação da estrutura de linguagem de alto nível é C++.

LOGFILE = *value*

O nome completo do z/OS UNIX do arquivo no qual DFHJS2LS grava seu log de atividades e suas informações de rastreamento. DFHJS2LS cria o arquivo, mas não a estrutura de diretório, se ela não existir.

Normalmente, você não usa esse arquivo, mas ele poderá ser solicitado pela organização de serviço IBM caso você encontre problemas com o DFHJS2LS.

MAPPING-LEVEL = { **1.0** | **1.1** | **1.2** | **2.0** | **2.1** | **2.2** | **3.0** | **4.0** | **4.1** }

Especifica o nível de mapeamento que o DFHJS2LS usa ao gerar o arquivo de ligação de serviço da web e a estrutura de linguagem. Você pode selecionar estas opções:

- 1.0** Essa opção é mantida para compatibilidade com serviços da web SOAP. Ela não é recomendada para uso com JSON.
- 1.1** Essa opção é mantida para compatibilidade com serviços da web SOAP. Ela não é recomendada para uso com JSON.
- 1.2** Essa opção é mantida para compatibilidade com serviços da web SOAP. Ela não é recomendada para uso com JSON.
- 2.0** Essa opção é mantida para compatibilidade com serviços da web SOAP. Ela não é recomendada para uso com JSON.

- 2.1 Essa opção é mantida para compatibilidade com serviços da web SOAP. Ela não é recomendada para uso com JSON.
- 2.2 Essa opção é mantida para compatibilidade com serviços da web SOAP. Ela não é recomendada para uso com JSON.
- 3,0 Use este nível de mapeamento para gerar o esquema JSON usando o conjunto completo de opções disponíveis.
- 4.0 Use este nível de mapeamento com uma região do CICS TS 5.2 quando você desejar usar UTF-16.
- 4.1 Para suporte de matriz truncável, use este nível de mapeamento com uma região do CICS TS 5.2 que tenha o APAR PI67641 aplicado ou qualquer região do CICS TS 5.3 ou mais recente.

Para obter mais informações sobre níveis de mapeamento, consulte Níveis de mapeamento para os assistentes CICS JSON.

MAPPING-OVERRIDES = { SAME-AS-MAPPING-LEVEL HYPHENS-AS-UNDERSCORES | | INTEGER-AS-PIC9 | LESS-DUP-NAMES | UNDERSCORES-AS-HYPHENS }

Especifica se o comportamento padrão é substituído para o nível de mapeamento especificado ao gerar estruturas de linguagem.

SAME-AS-MAPPING-LEVEL

Este parâmetro gera estruturas de linguagem no mesmo estilo que o nível de mapeamento. Este é o padrão.

HYPHENS-AS-UNDERSCORES

Para PL/I somente. Este parâmetro converte quaisquer hifens no esquema JSON em sublinhados em vez do caractere X, para melhorar a capacidade de leitura das estruturas de linguagem PL/I geradas. Para obter mais informações, consulte Esquema JSON para mapeamento de PL/I.

INTEGER-AS-PIC9

Apenas para COBOL e DFHJS2LS. Este parâmetro gera estruturas de linguagem que contêm valores de número inteiro do esquema JSON como numerais em vez de caracteres alfanuméricos.

LESS-DUP-NAMES

Este parâmetro gera nomes de campo de estrutura não estruturais com `_value` no final do nome para ativar a referência direta ao campo. Por exemplo, na estrutura de linguagem PLI a seguir, quando **MAPPING-OVERRIDES=LESS-DUP-NAMES** é especificado, o campo de nível 12 `streetName` é sufixado com `_value`:

```
09 streetName,
12 streetName CHAR(255) VARYING
UNALIGNED,
12 filler BIT (7),
12 attr_nil_streetName_value BIT (1),
```

A estrutura resultante é conforme a seguir:

```
09 streetName,
12 streetName_value CHAR(255) VARYING
UNALIGNED,
12 filler BIT (7),
12 attr_nil_streetName_value BIT (1),
```

UNDERSCORES-AS-HYPHENS

Esta opção é ativada automaticamente no nível de mapeamento 4.0.

Apenas para COBOL. Este parâmetro converte quaisquer sublinhados no esquema JSON em hifens, em vez do caractere X, para melhorar a capacidade de leitura das estruturas de linguagem COBOL geradas. Se ocorrerem quaisquer conflitos de nomes de campo, os campos serão numerados para garantir que sejam exclusivos. Para obter mais informações, consulte Esquema JSON para mapeamento COBOL.

MINIMUM-RUNTIME-LEVEL = { MINIMUM | 1.0 | 1.1 | 1.2 | 2.0 | 2.1 | 2.2 | 3.0 | 4.0 | 4.1 | **CURRENT** }

Especifica o ambiente de tempo de execução mínimo do CICS no qual o arquivo de ligação de serviço da web pode ser implementado. Se você selecionar um nível que não corresponde aos outros parâmetros que você especificou, receberá uma mensagem de erro. É possível selecionar essas opções:

MINIMUM

O menor nível de tempo de execução do CICS possível é alocado automaticamente conforme os parâmetros que você selecionou.

- 1.0** Essa opção é mantida para compatibilidade com serviços da web SOAP. Ela não é recomendada para uso com JSON.
- 1.1** Essa opção é mantida para compatibilidade com serviços da web SOAP. Ela não é recomendada para uso com JSON.
- 1.2** Essa opção é mantida para compatibilidade com serviços da web SOAP. Ela não é recomendada para uso com JSON.
- 2.0** Essa opção é mantida para compatibilidade com serviços da web SOAP. Ela não é recomendada para uso com JSON.
- 2.1** Essa opção é mantida para compatibilidade com serviços da web SOAP. Ela não é recomendada para uso com JSON.
- 2.2** Essa opção é mantida para compatibilidade com serviços da web SOAP. Ela não é recomendada para uso com JSON.
- 3.0** O arquivo de ligação de serviço da web gerado é implementado em uma região do CICS TS 4.1 ou mais recente.

Nota: O suporte JSON está disponível somente a partir do CICS TS 4.2 em diante.

- 4.0** O arquivo de ligação de serviço da web gerado é implementado com sucesso em um CICS TS 5.2 ou em uma região posterior. Com este nível de tempo de execução, é possível usar um nível de mapeamento 4.0 ou anterior para o parâmetro **MAPPING-LEVEL**. É possível utilizar qualquer parâmetro opcional neste nível.
- 4.1** O arquivo de ligação de serviço da web gerado é implementado com sucesso em uma região CICS TS V5.2 que tem o APAR PI67641 aplicado ou em qualquer região CICS TS V5.3 ou mais recente. Com este nível de tempo de execução, é possível utilizar um nível de mapeamento 4.1 ou anterior para o parâmetro **MAPPING-LEVEL**.

CURRENT

O arquivo de ligação de serviço da web gerado é implementado com sucesso em uma região CICS no mesmo nível de tempo de execução que você está usando para gerar o arquivo de ligação de serviço da web.

NAME-TRUNCATION = { **LEFT** | **RIGHT** }

Especifica se os nomes JSON são truncados a partir da esquerda ou da direita. O assistente de serviço da web do CICS trunca nomes JSON até o comprimento apropriado para a linguagem de alto nível especificada; por padrão, os nomes são truncados a partir da direita.

PDSCP = *value*

Especifica a página de códigos que é usada nos membros do conjunto de dados particionados que são especificados no parâmetro **PDSMEM**, em que *value* é um número de CCSID ou um número da página de códigos Java. Se este parâmetro não for especificado, a página de códigos do z/OS UNIX System Services será usada. Por exemplo, você pode especificar **PDSCP** = 037.

PDSLIB = *value*

Especifica o nome do conjunto de dados particionados que contém a linguagem de alto nível gerada. Os membros do conjunto de dados que são usados para a solicitação e a resposta são especificados no parâmetro **PDSMEM**.

PDSMEM = *value*

Especifica o prefixo de 1 a 6 caracteres que o DFHJS2LS usa para gerar o nome do membro do conjunto de dados particionados que conterá as estruturas de linguagem de alto nível.

DFHJS2LS gera um membro do conjunto de dados particionados anexando 01 ao prefixo.

PGMINT = **CHANNEL** | **COMMAREA**

Para um provedor de serviços, especifica como o CICS transmite dados ao programa de aplicativo de destino:

CHANNEL

CICS usa uma interface do canal para transmitir dados ao programa de aplicativo de destino.

COMMAREA

CICS usa uma área de comunicação para transmitir dados ao programa de aplicativo de destino.

Quando o programa de aplicativo de destino tiver processado a solicitação, ele deverá usar o mesmo mecanismo para retornar a resposta. Se a solicitação foi recebida em uma área de comunicação, a resposta deverá ser retornada na área de comunicação; se a solicitação foi recebida em um contêiner, a resposta deverá ser retornada em um contêiner. O comprimento da área de comunicação ou contêiner que o CICS transmite para o programa de aplicativo de destino é o maior dos comprimentos da área ou do contêiner de comunicação da solicitação e da área ou contêiner de comunicação de resposta.

PGMNAME = *value*

Especifica o nome de um recurso do PROGRAMA CICS.

Quando DFHJS2LS é usado para gerar um arquivo de ligação de serviço da web que é usado em um provedor de serviços, você deve fornecer esse parâmetro. Ele especifica o nome do recurso do programa de aplicativo que é exposto como um serviço da web.

Quando DFHJS2LS for usado para gerar um arquivo de ligação de serviço da web que é usado em um solicitante de serviço, omita este parâmetro.

STRUCTURE = *name*

Apenas para C e C++, especifica como o nome da estrutura é gerado.

A estrutura gerada recebe o nome *name01*.
Se o nome for omitido, a estrutura terá o mesmo nome que o nome do membro do conjunto de dados particionados gerado a partir do parâmetro **PDSMEM** que você especificar.

SYNCONRETURN = { NO | YES }

Especifica se o serviço da web remoto pode emitir um ponto de sincronização.

NO O serviço da web remoto não pode emitir um ponto de sincronização. Este valor é o padrão. Se o serviço da web remoto emitir um ponto de sincronização, ele falhará com um encerramento anormal de ADPL.

YES O serviço da web remoto pode emitir um ponto de sincronização. Se você selecionar YES, a tarefa remota é confirmada como uma unidade de trabalho separada quando o controle retorna do serviço da web remoto. Se o serviço da web remoto atualizar um recurso recuperável e ocorrer uma falha após ele retornar, a atualização para esse recurso não poderá ser restaurada.

TRANSACTION = *name*

Em um provedor de serviços, este parâmetro especifica um nome com um a quatro caracteres de uma transação de alias que pode iniciar o pipeline. O valor desse parâmetro é usado para definir o atributo TRANSACTION do recurso URIMAP quando ele é criado automaticamente usando o comando de varredura **PIPELINE**.

Caracteres aceitáveis:

A-Z a-z 0-9 \$ @ # _ < >

URI = *value*

Em um provedor de serviços, este parâmetro especifica o URI relativo que um cliente usa para acessar o serviço da web. O CICS usa o valor que é especificado quando ele gera um recurso URIMAP a partir do arquivo de ligação de serviço da web que é criado pelo DFHJS2LS. O parâmetro especifica o componente de caminho do URI ao qual a definição de URIMAP se aplica. Ao usar os curingas * no final de um URI, o valor de URI deve ser seguido por uma vírgula.

USERID = *id*

Em um provedor de serviços, este parâmetro especifica um ID do usuário de um a oito caracteres, o qual pode ser usado por qualquer Web client. Para uma resposta gerada por aplicativo ou um serviço da web, a transação de alias é conectada com este ID do usuário. O valor desse parâmetro é usado para definir o atributo USERID do recurso URIMAP quando ele é criado automaticamente usando o comando de varredura **PIPELINE**.

Caracteres aceitáveis:

A-Z a-z 0-9 \$ @ #

WIDE-COMP3 = { FULL | NO | YES }

Controla o tamanho máximo do comprimento variável decimal compactado na estrutura de linguagem COBOL ou PL/I gerada.

FULL Para COBOL e PL/I. DFHJS2LS gera um campo decimal compactado que é grande o suficiente para conter todos os valores válidos. O tamanho máximo são 31 dígitos. Este é o padrão.

NO Apenas para COBOL. DFHJS2LS limita o comprimento variável

decimal compactado a 18 ao gerar a estrutura de linguagem COBOL tipo COMP-3. Se o tamanho decimal compactado for maior que 18, a mensagem DFHPI9022W será emitida para indicar que o tipo especificado está sendo limitado a um total de 18 dígitos.

YES Apenas para COBOL. DFHJS2LS suporta o tamanho máximo de 31 ao gerar a estrutura de linguagem COBOL tipo COMP-3.

Nota: As opções NO e YES geram campos que são incapazes de representar todos os valores válidos; a opção FULL evita esse problema. No entanto, a opção FULL permite que alguns valores inválidos sejam representados no campo decimal compactado. Por exemplo, se um esquema indica que há um máximo de cinco dígitos e um máximo de dois dígitos fracionários, a opção FULL gerará um campo decimal compactado que permite sete dígitos e isso permite espaço para valores válidos como 25000 e 999,99, mas também fornece espaço para alguns valores inválidos como 9999,99. Quando você usar a opção FULL, tome cuidado para não gerar valores inválidos em dados do aplicativo.

WSBIND = *value*

O nome completo do z/OS UNIX do arquivo de ligação de serviço da web. DFHJS2LS cria o arquivo, mas não a estrutura de diretório, se ela não existir. A extensão do arquivo é padronizada como .wsbind.

Outras informações

- O ID do usuário sob o qual DFHJS2LS é executado deve ser configurado para usar o UNIX System Services. O ID do usuário deve ter permissão de leitura para a estrutura de arquivo e bibliotecas PDS do CICS z/OS UNIX e permissão de gravação para os diretórios especificados nos parâmetros **LOGFILE**, **WSBIND** e **WSDL**.
- O ID do usuário deve ter uma alocação de armazenamento suficientemente grande para executar Java.
- A JCL possui um comprimento de parâmetro máximo de 100 caracteres. Isso pode ser aumentado usando a instrução **STDPARM**. Para obter mais informações, consulte *z/OS UNIX System Services User Guide*.

Exemplo

```
//JS2LS JOB '  
accounting information  
'  
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/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  
/*
```

Criando um aplicativo do provedor de serviços JSON usando o assistente JSON

É possível criar um aplicativo do provedor de serviços a partir de um esquema JSON que está em conformidade com o esquema JSON v4 (rascunho) ou a partir de uma estrutura de dados de linguagem de alto nível. O assistente CICS JSON ajuda a implementar seus aplicativos CICS em uma configuração do provedor de serviços.

Sobre Esta Tarefa

Quando você usa o assistente para implementar um aplicativo CICS como um provedor de serviços, existem duas opções:

- Iniciar com um esquema JSON e usar o assistente para gerar as estruturas de dados de linguagem.
Usar esta opção quando desejar implementar um provedor de serviços que esteja em conformidade com uma descrição de serviços da web existente.
- Comece com as estruturas de dados de linguagem e use o assistente para gerar o esquema JSON.
Use esta opção quando desejar expor um programa existente como um serviço JSON.

Criando um aplicativo do provedor de serviços a partir de um esquema JSON

Usando o assistente CICS JSON, é possível criar um aplicativo do provedor de serviços a partir de um esquema JSON.

Antes de Iniciar

Antes de poder criar um aplicativo do provedor de serviços, as condições a seguir devem ser satisfeitas:

- Sua descrição de serviços da web deve estar em um arquivo UNIX no z/OS e você deve criar um pipeline do modo de provedor adequado na região do CICS.
- Deve-se definir para OMVS o ID do usuário sob o qual DFHJS2LS é executado.
- O ID do usuário deve ter permissão de leitura para z/OS UNIX e bibliotecas PDS e permissão de gravação para os diretórios especificados nos parâmetros **LOGFILE**, **WSBIND** e **JSON-SCHEMA-REQUEST**, **JSON-SCHEMA-RESPONSE** e **JSON-SCHEMA-RESTFUL**.
- Você deve alocar armazenamento suficiente para o ID do usuário para que o ID execute Java. É possível usar qualquer versão suportada de Java. Por padrão, DFHJS2LS usa a versão de Java especificada no parâmetro **JAVADIR**.

Sobre Esta Tarefa

É possível usar o assistente JSON para criar estruturas de linguagem a partir de seu esquema JSON para o aplicativo do provedor de serviços.

Procedimento

1. Use o programa em lote DFHJS2LS para gerar um arquivo de ligação de serviço da web e uma ou mais estruturas de dados de linguagem. O DFHJS2LS contém um grande conjunto de parâmetros opcionais que fornecem a flexibilidade para criar o arquivo de ligação e estruturas de linguagem que seu aplicativo requer. Considere essas opções quando ativar um aplicativo existente para serviços da web:

- Qual mecanismo o CICS usará para transmitir dados para o programa de aplicativo do provedor de serviços? É possível usar canais e transmitir os dados em contêineres ou usar uma COMMAREA. Os canais e contêineres são recomendados. Especifique-os com o parâmetro **PGMINT**.
- Qual linguagem deseja gerar? O DFHJS2LS pode gerar COBOL, C/C++ ou estruturas de dados de linguagem PL/I. Especifique a linguagem usando o parâmetro **LANG**.
- Qual nível de mapeamento deseja usar? Quanto maior o nível de mapeamento, mais controle e apoio você tem disponíveis para a manipulação de dados de caracteres e binários no tempo de execução. Alguns parâmetros opcionais estão disponíveis somente nos níveis de mapeamento mais altos. É recomendável usar o nível mais alto de mapeamento disponível. Especifique o nível de mapeamento com o parâmetro **MAPPING-LEVEL**.
- Qual URI deseja que o solicitante de serviço da web use? Especifique um URI relativo usando o parâmetro **URI**; por exemplo, URI=/my/test/webservice. O valor é usado pelo CICS quando ele cria o recurso URIMAP.
- Sob qual transação e ID do usuário você executará a solicitação e a resposta de serviço da web? É possível usar uma transação de alias para executar o aplicativo para compor uma resposta para o solicitante de serviço. A transação de alias é conectada sob o ID do usuário. Especifique-a com os parâmetros **TRANSACTION** e **USERID**. Estes valores são usados ao criar o recurso URIMAP. Se você não desejar usar uma transação específica, não use esses parâmetros.

Ao enviar DFHJS2LS, o CICS gera o arquivo de ligação de serviço da web e o coloca no local que você especificou com o parâmetro **WSBIND**. As estruturas de linguagem são colocadas no conjunto de dados particionados que você especificou com o parâmetro **PDSLIB**.

2. Copie o arquivo de ligação de serviço da web gerado no diretório pickup do recurso PIPELINE do modo de provedor que você deseja usar para seu aplicativo de serviço da web. Deve-se copiar o arquivo de ligação no modo binário.
3. Grave um programa de aplicativo do provedor de serviços na interface com as estruturas de linguagem geradas e implemente a lógica de negócios necessária.
4. Use o comando **PIPELINE SCAN** para criar dinamicamente o recurso WEBSERVICE e um recurso URIMAP.
 - O recurso WEBSERVICE encapsula o arquivo de ligação de serviço da web no CICS e é usado no tempo de execução.
 - O recurso URIMAP fornece ao CICS as informações para associar o recurso WEBSERVICE a um URI específico.

Como alternativa, é possível definir os recursos você mesmo, embora isso não seja recomendado.

Resultados

Se você tiver algum problema ao enviar DFHJS2LS, ou se os recursos não forem instalados corretamente, consulte Resolução de Problemas do assistente JSON.

Criando um Aplicativo do Provedor de Serviços a Partir de uma Estrutura de Dados

Use o assistente de serviços da web do CICS para criar um aplicativo do provedor de serviços a partir de uma estrutura de dados de linguagem de alto nível.

Antes de Iniciar

Antes de criar um aplicativo do provedor de serviços, certifique-se de que sua configuração atenda a essas pré-condições:

- Suas estruturas de dados de linguagem de alto nível devem atender aos seguintes critérios:
 - As estruturas de dados devem ser definidas separadamente do programa de origem; por exemplo, em um copybook COBOL.
 - Se o seu programa de aplicativo PL/I ou COBOL usar estruturas de dados diferentes para entrada e saída, as estruturas de dados deverão ser definidas em dois membros diferentes em um conjunto de dados particionados. Se a mesma estrutura for usada para entrada e saída, a estrutura deverá ser definida em um único membro.
- Para C e C++, suas estruturas de dados podem estar no mesmo membro em um conjunto de dados particionados.
- As estruturas de linguagem devem estar disponíveis em um conjunto de dados particionados e você deve criar um recurso PIPELINE adequado na região CICS.
- Deve-se definir para OMVS o ID do usuário que o DFHLS2JS utiliza para executar.
- O ID do usuário deve ter permissão de leitura para o z/OS UNIX e as bibliotecas PDS e permissão de gravação para os diretórios especificados nos parâmetros de saída **LOGFILE**, **WSBIND**, **JSON-SCHEMA-REQUEST** e **JSON-SCHEMA-RESPONSE**.
- O ID do usuário deve ter uma alocação de armazenamento suficientemente grande para executar Java. É possível usar qualquer versão suportada de Java. Por padrão, DFHLS2JS usa a versão de Java que é especificada no parâmetro **JAVADIR**.

Procedimento

Siga estas etapas para criar um aplicativo provedor de serviços a partir de uma estrutura de dados de alto nível:

1. Se a interface de aplicativo do provedor de serviços usar canais e muitos contêineres, crie um documento de descrição de canal que descreva a interface em JSON. Deve-se salvar o documento de descrição de canal em um diretório adequado no z/OS UNIX. O CICS usa este documento para construir e desconstruir uma mensagem JSON a partir dos contêineres em um canal. Como alternativa, é possível usar um contêiner em um canal e não criar um documento de descrição de canal.

Para obter mais informações sobre como criar um documento de descrição de canal, consulte “Criando um Documento de Descrição de Canal” na página 276.
2. Use o programa em lote DFHLS2JS para gerar um arquivo de ligação de serviço da web e a descrição de serviços da web da estrutura de linguagem. O programa em lote DFHLS2JS pode ser localizado em *HLQ.XDFHINST* em que *HLQ* é o local no qual você instalou o CICS. O DFHLS2JS contém um grande conjunto de parâmetros opcionais que fornecem a flexibilidade para criar o arquivo de ligação e as estruturas de linguagem que seu aplicativo requer. Considere as opções a seguir ao ativar serviços da web para um aplicativo existente:
 - Qual mecanismo você deseja que o CICS use para transmitir dados para o programa de aplicativo do provedor de serviços? É possível usar canais e transmitir os dados em contêineres ou usar uma COMMAREA. Especifique o mecanismo usando o parâmetro **PGMINT**. Se sua interface de aplicativo usar

canais e muitos contêineres, especifique o parâmetro **REQUEST-CHANNEL** e, opcionalmente, o **RESPONSE-CHANNEL**. É possível usar esses parâmetros somente quando o nível de mapeamento é 3.0 ou superior.

- Qual nível de mapeamento deseja usar? Quanto maior o nível de mapeamento, mais controle e apoio você tem disponíveis para a manipulação de dados de caracteres e binários no tempo de execução. Alguns parâmetros opcionais estão disponíveis somente nos níveis de mapeamento mais altos. Deve-se especificar o nível mais alto de mapeamento disponível no parâmetro **MAPPING-LEVEL**.
- Qual URI você deseja que o serviço da web use? Especifique um URI absoluto usando o parâmetro **URI**; por exemplo, **URI** = `http://www.example.org:80/my/test/webservice`. A parte relativa deste endereço, `/my/test/webservice`, é usada ao criar o recurso URIMAP.

Ao enviar DFHLS2JS, o CICS gera o arquivo de ligação de serviço da web e o coloca no local que você especificou com o parâmetro **WSBIND**. Os esquemas JSON gerados são colocados no local que você especificou com os parâmetros **JSON-SCHEMA-REQUEST** e **JSON-SCHEMA-RESPONSE**.

3. Revise o esquema JSON gerado.

Esses esquemas são usados para definir os formatos de dados de entrada e saída para o serviço da web JSON. O desenvolvedor de aplicativos deve usar esses esquemas ao criar um aplicativo para interagir com o serviço da web JSON.

Nota: A mudança no esquema gerado invalida o arquivo de ligação de serviço da web gerado, **WSBind**.

Se você deseja mudar o esquema, por exemplo, para renomear os campos dentro do esquema, deve usar o DFHJS2LS para gerar um novo arquivo de ligação de serviço da web e um novo conjunto de estruturas de linguagem. O programa de aplicativo no CICS deve ser mudado para usar as novas estruturas de linguagem.

4. Copie o arquivo de ligação de serviço da web no diretório de recebimento do pipeline do modo de provedor que você deseja usar para seu aplicativo de serviço da web. Deve-se copiar o arquivo de ligação de serviço da web no modo binário.
5. Use o comando **PIPELINE SCAN** para criar dinamicamente o recurso WEBSERVICE e um recurso URIMAP.
 - O recurso WEBSERVICE contém o arquivo de ligação de serviço da web no CICS e é usado no tempo de execução.
 - O recurso URIMAP fornece ao CICS as informações para associar o recurso WEBSERVICE a um URI específico.

Como alternativa, é possível definir os recursos você mesmo.

Resultados

A criação de seu aplicativo do provedor de serviços está concluída.

Se você tiver algum problema ao enviar o DFHLS2JS, ou se os recursos não forem instalados corretamente, consulte Resolução de Problemas do assistente JSON.

O que Fazer Depois

Disponibilize a descrição de serviços da web para qualquer pessoa que desenvolva um serviço da web que acessará seu serviço.

Criando um Documento de Descrição de Canal

Crie um documento de descrição de canal quando seu aplicativo do provedor de serviços utilizar uma interface de canal com vários contêineres.

Sobre Esta Tarefa

Use um editor XML para criar o documento de descrição de canal. O esquema para a descrição de canal é chamado de `channel.xsd` e está no diretório `/usr/lpp/cicsts/cicsts54 /schemas/channel` (em que `/usr/lpp/cicsts/ cicsts54` é o diretório de instalação padrão para arquivos CICS no z/OS UNIX).

Procedimento

1. Crie um documento XML com um elemento `<channel>` e o namespace do canal CICS:

```
<channel name="myChannel"
xmlns="http://www.ibm.com/xmlns/prod/CICS/channel">
</channel>
```

2. Inclua um elemento `<container>` para cada contêiner que a interface do programa de aplicativo usa no canal. Deve-se usar os atributos de nome, tipo e uso para descrever cada contêiner. O exemplo a seguir mostra seis contêineres com valores de atributo diferentes:

```
<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>
```

O elemento de estrutura indica que o conteúdo está definido em uma estrutura de linguagem localizada em um membro do conjunto de dados particionados.

3. Salve o documento XML no z/OS UNIX.

Esquema do Canal

O documento de descrição de canal deve estar em conformidade com o esquema a seguir:

```
<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 um canal do CICS.
2. Este elemento representa um contêiner do CICS dentro do canal.
3. Uma estrutura pode ser usada somente com contêineres do modo 'bit'. O atributo 'location' indica o local de um arquivo que mapeia o conteúdo do contêiner. O atributo 'structure' pode ser usado em C e C++ para indicar o nome da estrutura.

O que Fazer Depois

Execute DFHLS2JS para criar os mapeamentos e o esquema JSON para o aplicativo do provedor de serviços da web. O DFHLS2JS coloca os mapeamentos para o canal no esquema JSON na ordem em que os contêineres são especificados no documento de descrição de canal.

Customizando esquemas JSON gerados

Os esquemas JSON que são gerados por DFHLS2JS contêm algum conteúdo gerado automaticamente que pode ser apropriado mudar antes da publicação. A customização de esquemas JSON pode resultar na nova geração do arquivo de ligação de serviços da web e, em alguns casos, na gravação de um programa wrapper.

Sobre Esta Tarefa

Siga estas etapas para customizar os esquemas JSON gerados:

Procedimento

1. Para advertir o suporte para HTTPS, use o parâmetro **URI** no DFHLS2JS para configurar um URI absoluto.
2. Para fornecer o local de rede de seu serviço da web, use o parâmetro **URI** no DFHLS2JS para configurar um URI absoluto.
3. Considere se os nomes gerados automaticamente no esquema JSON são apropriados para seus propósitos. É possível renomear as propriedades.

Esses valores formam parte da interface programática na qual você codifica um programa cliente. Se os nomes gerados não são suficientemente significativos, a manutenção de seu código do aplicativo pode ser mais difícil durante um longo período de tempo.

Se você mudar qualquer um desses valores, deverá usar DFHJS2LS para gerar novamente o arquivo de ligação de serviços da web. As estruturas de linguagem que são produzidas provavelmente não são compatíveis com seu aplicativo existente, portanto, mudanças no aplicativo podem ser necessárias. Revise as estruturas de linguagem geradas e considere gravar um programa wrapper, conforme discutido na etapa 4.

4. Considere se os campos de COMMAREA expostos nos esquemas JSON são apropriados. Você pode considerar remover quaisquer campos que não são úteis para um desenvolvedor de cliente JSON:

- Os campos que são usados somente para valores de saída podem ser removidos do esquema que mapeia as estruturas de dados de entrada.
- Campos de preenchimento.
- Anotações geradas automaticamente.

Se você fizer alguma dessas mudanças, deverá gerar novamente o arquivo de ligação de serviços da web usando DFHJS2LS. As novas estruturas de linguagem que são geradas não são compatíveis com as estruturas de linguagem originais, portanto, você deve gravar um programa wrapper para mapear dados da nova representação para a antiga. Este programa wrapper precisa executar um comando **EXEC CICS LINK** para o programa de aplicativo de destino e, em seguida, mapear os dados retornados.

Este nível de customização requer o maior esforço, mas resulta em interfaces programáticas mais significativas para seus desenvolvedores de cliente JSON.

Resultados

Você tem um esquema JSON customizado que corresponde aos requisitos de negócios e um PROGRAM no CICS que os implementa.

Criando um aplicativo do provedor de serviços da web RESTful

A implementação do CICS de serviços da web JSON RESTful é semelhante à de serviços da web SOAP. A maioria dos conceitos e arquiteturas é compartilhada, mas o CICS requer o uso de um esquema JSON.

Sobre Esta Tarefa

A implementação de um serviço da web JSON RESTful envolve as seguintes tarefas:

Procedimento

1. Gerar a interface de aplicativo.

Entrada:

- O esquema JSON que define o modelo de dados para o serviço da web RESTful.

Saída:

- As estruturas de linguagem (por exemplo, copybook COBOL) que mapeiam o esquema JSON.
- Um arquivo WSBind.

Execute o utilitário DFHJS2LS e especifique os parâmetros de entrada apropriados. Esses parâmetros incluem:

- O local do esquema JSON.
- A lista de métodos suportados (GET, PUT, POST e DELETE são ativados por padrão).
- O URI no qual o serviço é implementado.
- O nome do PROGRAM do aplicativo que implementa o serviço.
- Quaisquer parâmetros de mapeamento de dados necessários.

2. Crie um aplicativo que utiliza essa interface.

Entrada:

- As estruturas de linguagem da etapa 1 na página 278.
- Um reconhecimento de quais operações RESTful serão implementadas

Saída:

- Um programa adequado para implementação no CICS

Grave um programa que faz o seguinte:

Nota: Se você fornecer o nome de seu próprio contêiner no parâmetro **CONTID** para DFHJS2LS, deverá usar este contêiner em vez de DFHWS-DATA sempre que ele for mencionado nas etapas a seguir.

- a. Examine o URI para entender a identidade do recurso. O CICS fornece vários contêineres para ajudá-lo a identificar componentes interessantes do URI. Os contêineres são descritos em Tabela 15. Os exemplos mostram o conteúdo de cada contêiner para o URI:

<http://www.example.org:10000/JSONServices/CustomerDetails/13388?action=query>

Se o URIMAP que correspondeu possui um PATH igual a
/JSONServices/CustomerDetails/ *

Tabela 15. Contêineres DFHWS-URI. contêineres DFHWS-URI

Contêiner	Conteúdo	Exemplo
DFHWS-URI	O URI completo	http://www.example.org:10000/JSONServices/CustomerDetails/13388?action=query
	A parte do caminho do URI que correspondeu ao URIMAP	/JSONServices/CustomerDetails/*
DFHWS-URI-RESID	O caminho do URI com a parte correspondida pelo URIMAP removido (o identificador de recurso)	13388
DFHWS-URI-QUERY	A sequência de consultas	action=query

- b. Valide o URI. Se houver um problema, relate o problema ao CICS e finalize.
- c. Consulte o contêiner DFHHTTPMETHOD para determinar qual método está sendo conduzido. Para obter mais informações, consulte DFHHTTPMETHOD.
- d. Para um método POST (create) (se necessário):
- Leia os dados de entrada a partir do contêiner DFHWS-DATA.

- Interprete os dados usando as estruturas de linguagem geradas na etapa 1 na página 278.
 - Valide os dados. Se houver um problema, relate o problema ao CICS e finalize.
 - Execute qualquer processamento específico do aplicativo necessário para criar o *recurso*.
 - Opcionalmente, grave no contêiner DFHRESPONSE para notificar o cliente do identificador quanto ao novo recurso. O conteúdo deste contêiner não é transformado pelo CICS, mas enviado diretamente na resposta de HTTP. O contêiner DFHWS-DATA é ignorado.
- e. Se um método GET (consultar) ou HEAD (consultar) for necessário, grave os dados que representam o contêiner *recurso* para o contêiner DFHWS-DATA.
- f. Se um método PUT (configurar) for necessário:
- Leia os dados de entrada a partir do contêiner DFHWS-DATA.
 - Interprete os dados usando as estruturas de linguagem geradas na etapa 1 na página 278.
 - Valide os dados. Se houver um problema, relate o problema ao CICS e finalize.
 - Execute qualquer processamento específico do aplicativo necessário para atualizar o *recurso*.
- g. Se um método DELETE for necessário, execute qualquer processamento específico do aplicativo necessário para excluir o *recurso*.

Nota: O modelo de dados RESTful que é implementado pelo CICS não envia um corpo de resposta para os métodos PUT, POST ou DELETE por padrão. Aplicativos RESTful geralmente usam o código de status HTTP para indicar sucesso ou falha. Se o aplicativo for concluído normalmente, o CICS enviará uma resposta de HTTP 200 (OK). Para obter mais informações sobre o envio de respostas de erro, consulte Relatório de erros de aplicativo. Se deseja enviar um corpo de resposta para um método PUT, POST ou DELETE, você deve gravar o contêiner DFHRESPONSE. Se presente, o CICS envia o conteúdo deste contêiner no corpo HTTP sem processamento adicional. O CICS ignora o contêiner DFHWS-DATA no processamento de resposta para esses métodos.

3. Implemente os artefatos.

Entrada:

- O arquivo WSBind da etapa 1 na página 278.
- O Programa CICS a partir da etapa 2 na página 279.
- Um recurso de provedor no modo de provedor do CICS que está configurado para JSON.

Saída:

- Um serviço da web JSON RESTful implementado.

Implemente o programa no CICS da maneira normal.

Ou:

- Implemente o arquivo WSBind no diretório de 'recebimento' do pipeline; em seguida, emita um comando **PIPELINE SCAN** para criar os recursos WEBSERVICE e URIMAP.
- Defina e instale manualmente um recurso WEBSERVICE e o recurso URIMAP associado. O URIMAP deve associar o URI ao PIPELINE e ao WEBSERVICE.

4. Teste o serviço.

Entrada:

- O esquema JSON.
- O URI do serviço da web RESTful.
- O serviço implementado na etapa 3.
- Um cliente de teste JSON de sua escolha.

Saída:

- Uma solicitação manipulada com sucesso.

Use o cliente de teste de sua escolha para enviar uma solicitação JSON ao CICS.

Se você receber uma resposta inesperada, tente a determinação de problemas. Para obter mais informações, consulte Resolvendo problemas com solicitações do JSON.

Considerações de design para aplicativos do provedor de serviços da web RESTful

Este tópico descreve alguns problemas que você deve considerar ao planejar e projetar um aplicativo do provedor de serviço da web RESTful para JSON.

Coleções de recursos

Um design comum para APIs RESTful é suportar a recuperação de coleções de recursos. Por exemplo, pode existir um Serviço que retorna um conjunto de objetos como a seguir:

```
GET /Services/CustomerDetails?Surname=Cooper
```

Esta solicitação deve retornar informações sobre todos os objetos CustomerDetails nos quais o Sobrenome é "Cooper". Os objetos CustomerDetails individuais podem ser retornados usando um URI mais específico tal como:

```
GET /Services/CustomerDetails/Customer27
```

Neste exemplo Customer27 é a chave primária para um cliente específico. A saída dessa segunda consulta será uma instância do objeto CustomerDetails. A saída da primeira consulta é menos clara: ela pode retornar uma lista de objetos CustomerDetails ou pode retornar uma lista de URIs para os objetos CustomerDetails (que o cliente pode continuar recuperando individualmente). Ambas as convenções são comuns.

Para implementar uma Coleção no CICS, crie um esquema JSON que descreva uma lista de instâncias de dados ou uma lista de URIs. É possível, então, construir o Serviço e implementá-lo normalmente. Neste exemplo, você pode optar por somente implementar o método GET. Você pode considerar a implementação de um Serviço de paginação para permitir que um cliente crie a página para trás e para frente por meio de grandes conjuntos de dados, por exemplo:

```
GET  
/Services/CustomerDetails?startRecord=200&endRecord=225
```

Você provavelmente precisará de dois recursos URIMAP no CICS (e dois recursos WEBSERVICE). Um que mapeia a estrutura do URI raiz para a Coleção e um URIMAP com curinga para as Instâncias. Por exemplo:

```
URIMAP1: Path=/Services/CustomerDetails  
WEBSERVICE=CollectionService  
URIMAP2: Path=/Services/CustomerDetails/* WEBSERVICE=InstanceService
```

Gerenciamento de cache

A arquitetura RESTful encoraja a integração com técnicas padrão de gerenciamento de cache HTTP. Isso permite que os resultados de solicitações GET sejam armazenados em cache na rede, reduzindo a carga no servidor. O mecanismo para fazer isto envolve a configuração de uma data/hora de expiração para os dados retornados para solicitações GET.

Não há nenhum mecanismo de propósito geral para suportar a expiração de cache controlado pelo aplicativo no CICS, mas um programa de Manipulador de Pipeline pode ser gravado para incluir o Cabeçalho HTTP apropriado usando EXEC CICS WEB WRITE HTTPHEADER API. Os programas de aplicativos podem fazer algo semelhante, mas somente se eles estão hospedados na mesma região CICS que recebe a solicitação de HTTP.

Matrizes variáveis de elementos no DFHJS2LS

O JSON pode conter matrizes de números variados de elementos. Em geral, Esquemas JSON que contêm números variados de elementos não mapeiam eficientemente para uma única estrutura de dados de linguagem de alto nível. O CICS usa mapeamentos baseados em contêiner ou mapeamentos sequenciais para manipular números variados de elementos em dados JSON.

Uma matriz com um número variado de elementos é representada no esquema JSON usando as palavras-chave `minItems` e `maxItems` no esquema com valor "type" igual a "array" :

- A palavra-chave `minItems` especifica o número mínimo de vezes que o elemento pode ocorrer. Ela pode ter um valor 0 ou qualquer número inteiro positivo. Ela é padronizada com o valor 0.
- A palavra-chave `maxItems` especifica o número máximo de vezes que o elemento pode ocorrer. Ele pode ter um valor de qualquer número inteiro positivo maior ou igual ao valor da palavra-chave `minItems`.
- Se a palavra-chave `maxItems` está ausente, significa que a matriz é ilimitada.

Um campo opcional pode ser indicado por uma matriz variável de `"maxItems":1`. Por exemplo, uma sequência opcional 8 bytes chamada "component" :

```
"properties":{
  "component": {
    "type":"array",
    "maxItems":1,
    "items":{
      "type" : "string",
      "maxLength": 8
    }
  },
  "required": ["component"]
```

O mesmo efeito pode ser produzido não incluindo o nome do campo no valor da palavra-chave "required":

```
"properties":{
  "component": {
    "type" : "string",
    "maxLength": 8
  }
}
```

Em geral, esquemas JSON que contêm números variados de elementos não mapeiam eficientemente para uma única estrutura de dados de linguagem de alto nível. Para lidar com esses casos, o CICS usa uma série de estruturas de dados conectadas que são transmitidas ao programa de aplicativo em uma série de contêineres. Essas estruturas são usadas como entrada e saída do aplicativo:

- Quando o CICS transforma dados JSON em dados do aplicativo, ele preenche estas estruturas com os dados do aplicativo e o aplicativo os lê.
- Quando o CICS transforma os dados do aplicativo em dados JSON, ele lê os dados do aplicativo nas estruturas que foram preenchidas pelo aplicativo.

Os exemplos a seguir ilustram o formato dessas estruturas de dados. Estes exemplos usam uma matriz de campos de 8 bytes simples. No entanto, o modelo suporta matrizes de tipos de dados complexos e matrizes de tipos de dados que contêm outras matrizes.

Exemplo 1. Número fixo de elementos

Este exemplo ilustra um elemento que ocorre exatamente três vezes:

```
"properties":{
  "component": {
    "type": "array",
    "maxItems": 3,
    "minItems": 3,
    "items":{
      "type" : "string",
      "maxLength": 8
    }
  },
  "required": ["component"]
```

Neste exemplo, o número de vezes que o elemento ocorre é conhecido antecipadamente, portanto, ele pode ser representado como uma matriz de comprimento fixo em uma declaração COBOL simples ou o equivalente em outras linguagens.

```
05 component PIC X(8) OCCURS 3 TIMES
```

Exemplo 2. Ativando o número de elementos no nível de mapeamento 2 e abaixo

Este exemplo ilustra um elemento obrigatório que pode ocorrer de uma a cinco vezes:

```
"properties":{
  "component": {
    "type": "array",
    "maxItems": 5,
    "minItems": 1,
    "items":{
      "type" : "string",
      "maxLength": 8
    }
  },
  "required": ["component"]
```

A estrutura de dados principal contém uma declaração de dois campos. Quando o CICS transforma os dados JSON em dados binários, o primeiro campo component-num contém o número de vezes que o elemento aparece nos dados JSON e o segundo campo, component-cont, contém o nome de um contêiner:

05 component-num PIC S9(9) COMP-5
05 component-cont PIC X(16)

Uma segunda estrutura de dados contém a declaração do elemento em si:

01 DFHJS-component
02 component PIC X(8)

Você deve examinar o valor de component-num, que conterá um valor no intervalo de 1 a 5, para descobrir quantas vezes o elemento ocorre. O conteúdo do elemento está no contêiner nomeado em component-cont; o contêiner contém uma matriz de elementos, em que cada elemento é mapeado pela estrutura de dados DFHJS-component.

Se minItems="0" , ou está ausente, e maxItems="1", o elemento é opcional. Para processar a estrutura de dados em seu programa de aplicativo, você deve examinar o valor de component-num:

- Se for zero, a mensagem não possui elemento do componente e o conteúdo de component-cont é indefinido.
- Se for um, o elemento do componente está no contêiner nomeado em component-cont.

O conteúdo do contêiner é mapeado pela estrutura de dados DFHJS-component.

Nota: Se os dados JSON consistirem em um único elemento recorrente, o DFHJS2LS gerará duas estruturas de linguagem. A estrutura de linguagem principal contém o número de elementos na matriz e o nome de um contêiner que contém a matriz de elementos. A segunda estrutura de linguagem mapeia uma única instância do elemento recorrente.

Exemplo 3. Número variado de elementos no nível de mapeamento 2.1 e superior

No nível de mapeamento 2.1 e acima, é possível usar o parâmetro **INLINE-MAXOCCURS-LIMIT** nos assistentes do CICS. O parâmetro **INLINE-MAXOCCURS-LIMIT** especifica a maneira como números variados de elementos são manipulados. As opções de mapeamento para números variados de elementos são o mapeamento baseado em contêiner, descrito em “Exemplo 2. Ativando o número de elementos no nível de mapeamento 2 e abaixo” na página 283 ou o mapeamento sequencial. O valor desse parâmetro pode ser um número inteiro positivo no intervalo de 0 a 32767:

- O valor padrão de **INLINE-MAXOCCURS-LIMIT** é 1 , o qual assegura que elementos opcionais sejam mapeados sequencialmente.
- Um valor 0 para o parâmetro **INLINE-MAXOCCURS-LIMIT** evita o mapeamento sequencial.
- Se maxItems for menor ou igual ao valor de **INLINE-MAXOCCURS-LIMIT**, o mapeamento sequencial será usado.
- Se maxItems for maior do que o valor de **INLINE-MAXOCCURS-LIMIT**, o mapeamento baseado em contêiner será usado.

O mapeamento de números variados de elementos sequenciais resulta na geração de uma matriz, como acontece com o exemplo de ocorrência fixa acima, e de um contador. O campo component-num indica quantas instâncias do elemento estão presentes e elas são apontadas pela matriz. Para o exemplo mostrado em “Exemplo

2. Ativando o número de elementos no nível de mapeamento 2 e abaixo” na página 283, quando **INLINE-MAXOCCURS-LIMIT** é menor ou igual a 5, a estrutura de dados gerada conforme a seguir:

```
05 component-num PIC S9(9) COMP-5 SYNC.  
05 component OCCURS 5 PIC X(8).
```

O primeiro campo, `component-num`, é idêntico ao de saída para o mapeamento baseado em contêiner de exemplo na seção anterior. O segundo campo contém uma matriz de comprimento 5 que é grande o suficiente para conter o número máximo de elementos que podem ser gerados.

O mapeamento sequencial difere do mapeamento baseado em contêiner, que armazena o número de ocorrências do elemento e o nome do contêiner no qual os dados são colocados, porque ele armazena todos os dados no contêiner atual. O armazenamento dos dados no contêiner atual geralmente irá melhorar o desempenho e tornar o mapeamento sequencial preferível.

Exemplo 4. Matrizes variáveis aninhadas

Os esquemas JSON complexos podem conter elementos variáveis recorrentes, que que por sua vez contêm elementos variáveis recorrentes. Nesse caso, a estrutura descrita se estende além dos dois níveis descritos nos exemplos.

Este exemplo ilustra um elemento opcional chamado "component2" que está aninhado em um elemento obrigatório chamado "component1", onde o elemento obrigatório pode ocorrer de uma a cinco vezes:

```
"properties":{  
  "component1": {  
    "type": "array",  
    "maxItems": 5,  
    "minItems": 1,  
    "items":{  
      "type": "object",  
      "properties":{  
        "component2":{  
          "type" : "string",  
          "maxLength": 8  
        }  
      },  
      "required": ["component2"]  
    }  
  },  
  "required": ["component1"]  
}
```

A estrutura de dados de nível superior é exatamente a mesma que nos exemplos anteriores:

```
05 component1-num PIC S9(9) COMP-5  
05 component1-cont PIC X(16)
```

No entanto, a segunda estrutura de dados contém estes elementos:

```
01 DFHJS-component1  
02 component2-num PIC S9(9) COMP-5  
02 component2-cont PIC X(16)
```

Uma estrutura de terceiro nível contém estes elementos:

```
01 DFHJS-component2  
02 component2 PIC X(8)
```

O número de ocorrências do elemento "component1" mais externo está em component1-num.

O contêiner nomeado em component1-cont contém uma matriz com esse número de instâncias da segunda estrutura de dados DFHJS-component1.

Cada instância de component2-cont nomeia um contêiner diferente, cada um dos quais contém a estrutura de dados mapeada pela estrutura de terceiro nível DFHJS-component2.

Para ilustrar essa estrutura, considere o fragmento de dados JSON que corresponde ao exemplo:

```
{ "component1":  
  [  
    {  
      "component2": "string1"  
    },  
    {  
      "component2": "string2"  
    },  
  ]  
}
```

"component1" ocorre três vezes. Os dois primeiros contêm uma instância de "component2", a terceira instância não.

Na estrutura de dados de nível superior, component1-num contém um valor igual a 3. O contêiner nomeado em component1-cont possui três instâncias de DFHJS-component1 :

1. Na primeira, component2-num possui um valor 1 e o contêiner nomeado em component2-cont contém *string1*.
2. Na segunda, component2-num possui um valor 1 e o contêiner nomeado em component2-cont contém *string2*.
3. Na terceira, component2-num possui um valor 0 e o conteúdo de component2-cont é indefinido.

Neste exemplo, a estrutura de dados completa é representada por quatro contêineres:

- A estrutura de dados raiz no contêiner DFHJS-DATA.
- O contêiner nomeado em component1-cont.
- Dois contêineres nomeados nas duas primeiras instâncias de component2-cont.

Estruturas opcionais e a palavra-chave **required**

Estruturas de dados são definidas pelo Esquema JSON "type" de "object". Os esquemas relacionam nomes de campos a tipos individuais usando o objeto fornecido pela palavra-chave "properties". O requisito para esses campos fazerem parte dos dados JSON descritos pelo Esquema JSON é controlado pela matriz fornecida pela palavra-chave "required". Essa matriz lista todos os nomes de campos que devem estar presentes nos dados JSON. Os campos opcionais são, portanto, representados por sua ausência nessa matriz ou, como a matriz não tem permissão para estar vazia, a ausência da palavra-chave "required". Nesse caso, todos os campos são opcionais.

Os campos opcionais são tratados como uma matriz variável de 0 ou 1 elemento. Isso inclui um campo adicional com o sufixo "-num" anexado ao nome de

elemento. Se o comprimento total é maior que 28 caracteres, o nome de elemento é truncado. No tempo de execução isso será diferente de zero para indicar que o valor estava presente nos dados JSON e zero se não estava.

Este exemplo mostra dois campos, um necessário chamado "required-structure" e o outro opcional chamado "optional-structure" :

```
{
  "type": "object",
  "properties": {
    "required-structure": {
      "type": "string",
      "maxLength": 8
    },
    "optional-structure": {
      "type": "string",
      "maxLength": 8
    }
  },
  "required": [
    "required-structure"
  ]
}
```

A estrutura COBOL gerada mostra ambos os campos, mas o segundo é precedido por "optional-structure-num" que é uma contagem de número inteiro dos elementos, com 0 representando nenhum e 1 indicando que ele está presente. O valor é configurado para indicar se o "optional-structure" contém dados válidos ou não.

```
03 OutputData.
06 required-structure PIC X(8).
06 optional-structure-num PIC S9(9) COMP-5 SYNC.
06 optional-structure PIC X(8).
```

Relatório de erros de aplicativo

Leia este tópico para entender como os aplicativos do provedor de serviço da web RESTful JSON podem relatar erros para os clientes.

Em cenários descendentes, é provável que o aplicativo seja requerido para relatar condições de erro. Por exemplo, um erro pode ser: "Número da Conta não reconhecido". Esse requisito é exclusivo para cenários descendentes; no desenvolvimento ascendente o aplicativo possui um mecanismo de relatório de erro que é codificado nos campos de dados ou ele encerra de forma anormal. No cenário descendente, o esquema JSON provavelmente não definirá um campo no qual relatar erros, então uma alternativa é necessária.

Para serviços da web baseados em SOAP, esse problema é tratado usando a API **EXEC CICS SOAPFAULT**. As mensagens de Falha de SOAP não existem no JSON. Em vez disso, é possível usar o contêiner DFHHTTPSTATUS para relatar erros detectados pelo aplicativo para aplicativos JSON. Para obter mais informações, consulte DFHHTTPSTATUS.

Nota: Os aplicativos também podem usar o contêiner DFHRESPONSE, e outros contêineres de controle, para fornecer uma resposta de erro mais detalhada, se eles desejam fazer isso.

Restrições de serviço da web JSON

Utilize esse material de referência para entender recursos que não são suportados pelos serviços da web JSON.

Os seguintes recursos não são suportados:

- Pipelines no modo do solicitante com JSON não são suportados.
- A validação de tempo de execução de dados JSON no esquema não é suportada. O valor do atributo VALIDATION de um recurso WEBSERVICE que é usado com uma carga útil JSON é ignorado.
- O uso de namespaces em dados JSON (convenções de Badgerfish ou Mapped) não é suportado.
- Cargas úteis JSON enviadas para o CICS devem ser codificadas em UTF-8. Nenhuma outra codificação é suportada. De modo semelhante, o JSON enviado pelo CICS é sempre codificado em UTF-8.
- Os transportes do WebSphere MQ com pipelines JSON não são suportados.
- Os programas transformadores do fornecedor não são suportados para uso com o transformador JSON.
- A reutilização de arquivos WSBIND que são criados para aplicativos de serviços da web SOAP em um pipeline JSON não é suportada. Arquivos WSBIND para uso com aplicativos do provedor de serviços JSON devem ser gerados pelo assistente JSON.
- Se uma carga útil JSON não possui algum dos conteúdos obrigatórios quando o CICS a transforma, os campos equivalentes dentro das estruturas de dados não são inicializados quando transmitidos ao programa de aplicativo.
- O CICS não pode transformar valores de número inteiro maiores do que o valor máximo para um longo sinalizado ($2^{63} - 1$), a menos que eles sejam colocados entre aspas.
- O uso de tipos de dados simples não é suportado na raiz de um esquema JSON. O esquema JSON descreve um objeto JSON ou uma matriz JSON, embora o objeto JSON possa, por sua vez, conter tipos de dados simples, matrizes e outros objetos.
- Se uma matriz é declarada em um esquema JSON com um valor **maxItems** de 1, o CICS serializa a matriz como uma sequência ou número inteiro simples ao gerar JSON no tempo de execução.

Importante: Os únicos caracteres suportados para nomes de propriedades JSON são: A-Z a-z _ : para o primeiro caractere e A-Z a-z 0-9 _ : . - para todos os caracteres subsequentes.

Atualmente, o suporte de serviços da web Axis2 tem inúmeras opções para desenvolvimento e implementação de aplicativos e customizações. As seguintes opções não são suportadas:

- Manipuladores de aplicativo fornecido pelo usuário - deve-se usar a classe do manipulador de aplicativo fornecida pelo CICS, `com.ibm.cicsts.axis2.CICSAXIS2ApplicationHandler`.
- Aplicativos Java Axis2 gravados pelo usuário.
- As APIs SOAPFAULT e WS-Addressing não podem ser usadas com o pipeline JSON.

Restrições do contêiner

Nota: Alguns contêineres de pipelines não são preenchidos quando uma solicitação JSON é processada. Para obter mais informações, consulte Contêineres usados no pipeline.

Diferenças em serviços da web RESTful

Em INQUIRE PIPELINE:

- SOAPLEVEL retorna NOTSOAP.
- Os atributos MTOMNOXOPST, MTOMST, SENDMTOMST, SOAPRNUM, SOAPVNUM, XOPDIRECTST e XOPSUPPORTST não são usados.

Em INQUIRE WEBSERVICE:

- Os atributos ARCHIVEFILE, BINDING, VALIDATIONST, XOPDIRECTST e XOPSUPPORTST não são usados.
- WSDLFILE retorna o nome do arquivo de esquema JSON que está associado ao WEBSERVICE.

No recurso WEBSERVICE:

- Os parâmetros ARCHIVEFILE e VALIDATION não são usados e seus valores são ignorados.
- WSDLFILE é o nome do arquivo de esquema JSON que está associado ao WEBSERVICE.

Mapeando e transformando dados do aplicativo e XML

É possível gravar programas de aplicativos para transformar dados binários do aplicativo em XML e vice-versa. O CICS suporta um número de linguagens de alto nível e fornece um assistente de XML para mapear como os dados são transformados durante o processamento do tempo de execução. O CICS usa a mesma tecnologia para mapear dados do aplicativo para XML em mensagens SOAP, como parte do suporte de serviços da web.

Antes de Iniciar

Deve-se ter o Java instalado para executar o assistente XML e, opcionalmente, executar a validação quando o CICS transforma os dados ou XML.

Sobre Esta Tarefa

A vantagem de usar essa abordagem para transformar dados do aplicativo em/do XML é que esse CICS vai além das capacidades oferecidas por um analisador XML. O CICS pode interpretar o XML e executar conversões baseadas em registro dos dados do aplicativo. Portanto, é mais fácil e mais rápido para você criar aplicativos que funcionam com o XML usando essa abordagem.

O assistente de XML do CICS é um utilitário fornecido que ajuda a criar os artefatos necessários para transformar dados binários do aplicativo em XML ou transformar XML em dados binários do aplicativo. O assistente de XML pode criar os artefatos em um diretório de pacote configurável ou em outro local especificado no z/OS UNIX.

Procedimento

1. Crie os mapeamentos usando o assistente XML.
2. Crie os recursos do XMLTRANSFORM no CICS para tornar os mapeamentos disponíveis.
3. Crie ou atualize um programa de aplicativo para usar o comando **TRANSFORM** API. O aplicativo deve usar uma interface baseada em canal.

4. Execute o aplicativo para testar se a transformação funciona conforme desejado. É possível ativar a validação para verificar se o CICS converte os dados corretamente.

O que Fazer Depois

Estas etapas são explicadas com mais detalhes nos tópicos a seguir.

O assistente XML do CICS

O assistente XML do CICS é um conjunto de utilitários em lote que podem ajudá-lo a transformar XML em estruturas de linguagem de alto nível e vice-versa. O assistente suporta a implementação rápida de aplicativos que executam processamento XML com a quantidade mínima de esforço de programação.

O uso do assistente XML para CICS reduz a quantidade de código que você deve gravar para analisar ou construir XML; o CICS transforma dados entre fragmentos XML e a estrutura de dados de um programa aplicativo.

O assistente XML pode criar um esquema XML a partir de uma estrutura de linguagem simples ou uma estrutura de linguagem a partir de um esquema XML existente e suporta COBOL, C/C++ e PL/I. Ele também gera metadados que o CICS usa no tempo de execução para converter automaticamente dados XML em dados binários do aplicativo ou vice-versa, os metadados são definidos em uma ligação XML e armazenados no z/OS UNIX. O esquema para a ligação XML está no diretório `/usr/lpp/cicsts/cicsts52/schemas/xmltransform/` no z/OS UNIX.

O assistente XML do CICS consiste em dois programas utilitários:

DFHLS2SC

Este utilitário gera um esquema XML e uma ligação a partir de uma estrutura de linguagem.

DFHSC2LS

Esse utilitário gera uma ligação XML e a estrutura de linguagem que você pode usar em seus programas de aplicativo. É possível usar um documento WSDL ou um esquema XML como entrada.

Os procedimentos JCL para executar ambos os programas estão na biblioteca `hlq.XDFHINST` em que `hlq` é o qualificador de alto nível de sua instalação do CICS.

DFHLS2SC: Linguagem de alto nível para conversão de esquema XML

O procedimento catalogado DFHLS2SC gera um esquema XML e um arquivo de ligação XML a partir de uma estrutura de linguagem de alto nível. Use DFHLS2SC quando desejar criar um programa CICS que possa analisar ou criar XML. As instruções de controle de tarefa para DFHLS2SC, seus parâmetros simbólicos, seus parâmetros de entrada e suas descrições são listados.

Instruções de Controle de Tarefa para DFHLS2SC

JOB Inicia a tarefa.

EXEC Especifica o nome do procedimento (DFHLS2SC).

INPUT.SYSUT1 DD

Especifica a entrada. Os parâmetros de entrada geralmente são

especificados no fluxo de entrada. No entanto, é possível defini-los em um conjunto de dados ou em um membro de um conjunto de dados particionados.

Parâmetros simbólicos

Os parâmetros simbólicos a seguir são definidos no DFHLS2SC:

JAVADIR = *path*

Especifica o nome do diretório Java que é usado pelo DFHLS2SC. O valor desse parâmetro é anexado a `/usr/lpp/` fornecendo um nome de caminho completo de `/usr/lpp/path`.

Normalmente, você não especifica este parâmetro. O valor padrão é o valor que foi fornecido à tarefa de instalação do CICS (DFHISTAR) no parâmetro **JAVADIR**.

PATHPREF = *prefix*

Especifica um prefixo opcional que estende o caminho do diretório z/OS UNIX usado em outros parâmetros. O padrão é a cadeia vazia.

Normalmente, você não especifica este parâmetro. O valor padrão é o valor que foi fornecido para a tarefa de instalação do CICS (DFHISTAR) no parâmetro **PATHPREF**.

SERVICE = *value*

Use este parâmetro somente quando orientado a fazer isso pelo suporte IBM.

TMPDIR = *tmpdir*

Especifica o local de um diretório no z/OS UNIX que o DFHLS2SC usa como uma área de trabalho provisória. O ID do usuário sob o qual a tarefa é executada deve ter permissão de leitura e gravação para esse diretório.

O valor padrão é `/tmp`.

TMPFILE = *tmpprefix*

Especifica um prefixo que o DFHLS2SC usa para construir os nomes dos arquivos da área de trabalho provisória.

O valor padrão é `SC2WS`.

USSDIR = *path*

Especifica o nome do diretório do CICS TS no sistema de arquivos do z/OS UNIX. O valor desse parâmetro é anexado a `/usr/lpp/cicsts/`, fornecendo um nome de caminho completo de `/usr/lpp/cicsts/path`.

Normalmente, você não especifica este parâmetro. O valor padrão é o valor que foi fornecido à tarefa de instalação do CICS (DFHISTAR) no parâmetro **USSDIR**.

O Espaço de Trabalho Temporário

DFHLS2SC cria os três arquivos temporários a seguir no tempo de execução:

```
tmpdir / tmpprefix .in  
tmpdir / tmpprefix .out  
tmpdir / tmpprefix .err
```

em que:

tmpdir é o valor especificado no parâmetro **TMPDIR**.

tmpprefix é o valor especificado no parâmetro **TMPFILE**.

```
/tmp/LS2SC.in
/tmp/LS2SC.out
/tmp/LS2SC.err
```

Portanto, você será avisado para planejar uma convenção de nomenclatura e procedimentos operacionais que evitem esta situação; por exemplo, é possível usar o parâmetro simbólico do sistema **SYSUID** para gerar nomes de arquivos da área de trabalho que sejam exclusivos para um usuário individual.

Parâmetros de Entrada para DFHLS2SC



Utilização de Parâmetros

- Você pode especificar os parâmetros de entrada em qualquer ordem.
- Cada parâmetro deve ser iniciado em uma nova linha.
- Um parâmetro (e seu caractere de continuação, se você usar um) não deve estender além da coluna 72; as colunas 73 a 80 devem conter espaços em branco.
- Se um parâmetro é muito longo para se ajustar em uma única linha, use um caractere de asterisco (*) no final da linha para indicar que o parâmetro continua na próxima linha. Tudo (incluindo espaços) antes do asterisco é considerado parte do parâmetro. Por exemplo:

```
XSDBIND=/path/xsdbindir*  
        /app1
```

é equivalente a

```
XSDBIND=/path/xsdbindir/app1
```

- Um caractere # na primeira posição de caractere da linha é um caractere de comentário. A linha é ignorada.
- Uma vírgula na posição de último caractere da linha é um separador de linha opcional e é ignorado.

Descrições de Parâmetros

BUNDLE = *value*

Especifica o caminho e o nome do diretório do pacote configurável no z/OS UNIX. Se você especificar esse valor, o assistente XML gerará um pacote configurável contendo a ligação XSD. As informações de caminho neste parâmetro substituem quaisquer informações de caminho no parâmetro **XSDBIND**.

Opcionalmente, é possível especificar um archive em vez de um nome de diretório. O assistente XML suporta archives .zip e .jar. No entanto, deve-se descompactar o archive antes de tentar instalar o recurso BUNDLE.

Se você não especificar esse parâmetro, o CICS colocará o esquema XML e a ligação no local especificado no parâmetro **XSDBIND**.

CCSID = *value*

Especifica o CCSID que é usado no tempo de execução para codificar dados de caractere na estrutura de dados do aplicativo. O valor desse parâmetro substitui o valor do parâmetro de inicialização do sistema **LOCALCCSID**. O *value* deve ser um CCSID EBCDIC que seja suportado pelos serviços de conversão Java e z/OS. Se você não especificar esse parâmetro, a estrutura de dados do aplicativo será codificada usando o CCSID especificado no parâmetro de inicialização do sistema.

Este parâmetro pode ser utilizado com qualquer nível de mapeamento.

CHAR-VARYING = { **NO** | **NULL** | **COLLAPSE** | **BINARY** }

Especifica como os campos de caractere na estrutura de linguagem são mapeados quando o nível de mapeamento é 1.2 ou superior. Um campo de caractere em COBOL é uma cláusula Picture do tipo X; por exemplo, PIC(X)10. Um campo de caractere em C/C++ é uma matriz de caracteres. Este parâmetro não se aplica às estruturas de linguagem PL/I Enterprise e Other. Você pode selecionar estas opções:

NO Os campos de caractere são mapeados para um `xsd:string` e são processados como campos de comprimento fixo. O comprimento

máximo dos dados é igual ao comprimento do campo. NO é o valor padrão para o parâmetro **CHAR-VARYING** para COBOL e PL/I nos níveis de mapeamento 2.0 e anteriores.

NULL Campos de caractere são mapeados para um `xsd:string` e são processados como sequências com terminação nula. O CICS inclui um caractere nulo de terminação ao transformar a partir de um esquema XML. O comprimento máximo da sequência de caracteres é calculado como um caractere menor que o comprimento indicado na estrutura de linguagem. NULL é o valor padrão para o parâmetro **CHAR-VARYING** para C/C++.

COLLAPSE

Os campos de caractere são mapeados para um `xsd:string`. Espaços em branco finais no campo não são incluídos no esquema XML. COLLAPSE é o *value* padrão para o parâmetro **CHAR-VARYING** para COBOL e PL/I no nível de mapeamento 2.1 em diante.

BINARY

Os campos de caractere são mapeados como um tipo de dados `xsd:base64binary` e são processados como campos de comprimento fixo. O BINARY *value* no parâmetro **CHAR-VARYING** está disponível somente no nível de mapeamento 2.1 e posteriores.

CHAR-OCCURS = { STRING | ARRAY }

Especifica como matrizes de caracteres na estrutura de linguagem são mapeados quando o nível de mapeamento é 4.0 ou superior. Por exemplo, PIC X OCCURS 20 . Este parâmetro é somente para uso pela linguagem COBOL.

ARRAY

Matrizes de caracteres são mapeadas para uma matriz XML. Isso significa que cada caractere é mapeado como um elemento XML individual. Este também é o comportamento nos níveis de mapeamento 3.0 e anteriores.

CADEIA

As matrizes de caracteres são mapeadas para uma sequência XML. Isso significa que a matriz COBOL inteira é mapeada como um elemento XML único.

CHAR-USAGE = { NATIONAL | DBCS }

Em COBOL, o tipo de dado nacional, PIC N, pode ser usado para dados UTF-16 ou DBCS. Essa configuração é controlada pela opção do compilador NSYMBOL. Deve-se configurar o parâmetro **CHAR-USAGE** no assistente com o mesmo valor que a opção do compilador NSYMBOL para assegurar que os dados sejam manipulados apropriadamente. Isso geralmente é configurado como CHAR-USAGE=NATIONAL quando você usa UTF-16.

DBCS Dados dos campos PIC (*n*) são tratados como dados codificados em DBCS.

NATIONAL

Dados dos campos PIC (*n*) são tratados como dados codificados em UTF-16.

DATA-SCREENING = { ENABLED | DISABLED }

Especifica se os dados fornecidos pelo aplicativo foram verificados com relação aos erros.

ENABLED

Quaisquer dados inconsistentes de tempo de execução fornecidos pelo

aplicativo com a estrutura de linguagem são tratados como um erro e a mensagem DFHPI1010 é emitida. Uma resposta de erro é retornada ao aplicativo.

DISABLED

Os valores dos dados de tempo de execução fornecidos pelo aplicativo que são inconsistentes com a estrutura de linguagem são substituídos pelos valores padrão. Por exemplo, um zero substitui um valor inválido em um campo numérico. A mensagem DFHPI1010 não é emitida e uma resposta normal é retornada ao aplicativo. Esse recurso pode ser usado para evitar as respostas de erro INVALID_PACKED_DEC e INVALID_ZONED_DEC que são geradas a partir de campos de saída não inicializados.

DATA-TRUNCATION = { **DISABLED** | **ENABLED** }

Especifica se os dados de comprimento variável são tolerados em uma estrutura de campo de comprimento fixo:

DISABLED

Se os dados forem menores do que o comprimento fixo que o CICS está esperando, o CICS rejeitará os dados truncados e emitirá uma mensagem de erro.

ENABLED

Se os dados forem menores do que o comprimento fixo que o CICS está esperando, o CICS tolerará os dados truncados e processará os dados ausentes como valores nulos.

DATETIME = { **UNUSED** | **PACKED15** }

Especifica se campos ABSTIME em potencial na estrutura de linguagem de alto nível são mapeados como registros de data e hora:

PACKED15

Os campos decimais compactados de comprimento 15 (8 bytes) são tratados como campos CICS ABSTIME e mapeados como registros de data e hora.

UNUSED

Campos decimais compactados de comprimento 15 (8 bytes) não são tratados como registros de data e hora.

Você pode configurar este parâmetro no nível de mapeamento 3.0.

LANG = **COBOL**

Especifica que a linguagem de programação da estrutura de linguagem de alto nível é COBOL.

LANG = **PLI-ENTERPRISE**

Especifica que a linguagem de programação da estrutura de linguagem de alto nível é Enterprise PL/I.

LANG = **PLI-OTHER**

Especifica que a linguagem de programação da estrutura de linguagem de alto nível é um nível de PL/I diferente de Enterprise PL/I.

LANG = **C**

Especifica que a linguagem de programação da estrutura de linguagem de alto nível é C.

LANG = **CPP**

Especifica que a linguagem de programação da estrutura de linguagem de alto nível é C++.

LOGFILE = *value*

O nome completo do arquivo do z/OS UNIX no qual o DFHLS2SC grava seu log de atividades e informações de rastreamento. O DFHLS2SC criará o arquivo, mas não a estrutura de diretório, se ele ainda não existir.

Normalmente, você não usa esse arquivo, mas ele poderá ser solicitado pela organização de serviço da IBM caso você encontre problemas com o DFHLS2SC.

MAPPING-LEVEL = { **1.0** | **1.1** | **1.2** | **2.0** | **2.1** | **2.2** | **3.0** | **4.0** | **4.1** }

Especifica o nível de mapeamento para o assistente usar ao gerar a ligação XML e as estruturas de linguagem. É recomendável usar o nível de mapeamento mais recente disponível. Se você estiver criando uma ligação XML para um feed Atom, deverá usar um nível de mapeamento 3.0.

MINIMUM-RUNTIME-LEVEL = { **MINIMUM** | **3.0** | **4.0** | | **4.1 CURRENT** }

Especifica o ambiente de tempo de execução mínimo do CICS no qual a ligação XML pode ser implementada. Se você selecionar um nível que não corresponde aos outros parâmetros que você especificou, você receberá uma mensagem de erro. As opções que podem ser selecionadas são as seguintes:

MINIMUM

O menor nível de tempo de execução possível do CICS é alocado automaticamente com base nos parâmetros que você especificou.

3,0 Especifique o nível de tempo de execução 3.0 ou acima se deseja usar o assistente CICS XML e tirar vantagem dos mapeamentos de dados avançados.

4.0 O arquivo de ligação de serviço da web gerado é implementado com sucesso em um CICS TS 5.2 ou em uma região posterior. Com este nível de tempo de execução, é possível usar um nível de mapeamento 4.0 ou anterior para o parâmetro **MAPPING-LEVEL**. É possível utilizar qualquer parâmetro opcional neste nível.

4.1 O arquivo de ligação de serviços da web gerado é implementado com sucesso em uma região CICS V5.2 que possui o APAR PI67641 aplicado ou em qualquer região CICS V5.3 ou mais recente. Com este nível de tempo de execução, é possível utilizar um nível de mapeamento 4.1 ou anterior para o parâmetro **MAPPING-LEVEL**.

CURRENT

Use este nível de tempo de execução para implementar o arquivo de ligação gerado em uma região CICS que possui o ambiente de tempo de execução configurado com aquele usado para gerar o arquivo de ligação.

NAMESPACE = *value*

Especifica o namespace para o CICS usar no esquema XML gerado. Para feeds Atom, o CICS fornece este namespace no feed Atom juntamente com o namespace de Atom.

Se você não especificar este parâmetro, o CICS gerará um namespace automaticamente.

OVERWRITE-OUTPUT = **NO** | **YES**

Controla se **BUNDLES** do CICS existentes no sistema de arquivos podem ser sobrescritos.

NO Qualquer **BUNDLE** existente não é substituído. Se um **BUNDLE** existente for localizado, o DFHLS2SC emitirá uma mensagem de erro DFHPI9689E e terminará.

YES Qualquer BUNDLE existente é substituído. Se um BUNDLE existente for localizado, a mensagem DFHPI9683W será emitida para informá-lo que o arquivo foi substituído.

PDSCP = *value*

Especifica a página de códigos que é usada nos membros do conjunto de dados particionados, em que *value* é um número de CCSID ou um número da página de códigos Java. Se você não especificar esse parâmetro, a página de códigos do z/OS UNIX System Services será usada. Por exemplo, você pode especificar PDSCP=037.

PDSLIB = *value*

Especifica o nome do conjunto de dados particionados que contém as estruturas de dados de linguagem de alto nível a serem processadas.

Restrição: Os registros no conjunto de dados particionados deve ter um comprimento fixo de 80 bytes.

PDSMEM = *value*

Especifica o nome do membro do conjunto de dados particionados que contém as estruturas de linguagem de alto nível a serem processadas.

SCHEMA = *value*

O nome completo do arquivo do z/O UNIX no qual o esquema XML é gravado. O esquema XML está em conformidade com a especificação WSDL 2.0. O DFHLS2SC criará o arquivo, mas não a estrutura de diretório, se ele ainda não existir.

STRUCTURE = { DFHDATA | *data* }

O nome da estrutura de dados de nível superior em C e C++. O padrão é DFHDATA.

TRUNCATE-NULL-ARRAYS = { DISABLED | ENABLED }

Especifica como matrizes estruturadas são processadas no nível de mapeamento 4.1 ou superior. Se ativado, o CICS tentará reconhecer registros vazios em uma matriz (consulte TRUNCATE-NULL-ARRAY-VALUES para obter mais informações sobre como identificar os registros vazios). Se cinco registros de matriz vazia consecutivos forem detectados, a matriz será truncada no primeiro registro ao gerar XML/JSON. Esse recurso de truncamento é ativado somente para matrizes com conteúdo estruturado, matrizes de campos primitivos simples não estão sujeitas a truncamento. O truncamento de matrizes pode resultar em uma representação mais concisa dos dados em JSON/XML, mas não é sem risco. Se cinco registros de dados consecutivos forem identificados incorretamente como armazenamento não inicializado (talvez porque eles legitimamente contêm valores baixos), poderá haver perda de dados. Se TRUNCATE-NULL-ARRAYS estiver ativado e TRUNCATE-NULL-ARRAY-VALUES não estiver configurado, o valor padrão para TRUNCATE-NULL-ARRAY-VALUES será usado.

TRUNCATE-NULL-ARRAY-VALUES = { NULL | SPACE | ZERO }

Especifica quais valores são tratados como vazios para processamento de TRUNCATE-NULL-ARRAYS no nível de mapeamento 4.1 ou superior. Por padrão, o valor nulo (0x00 ou valores baixos) é tratado como vazio. Se todos os bytes de armazenamento em um registro de uma matriz estruturada contêm valores nulos, o registro inteiro é considerado vazio. Um ou mais dos valores NULL, SPACE e ZERO podem ser especificados em uma lista separada por vírgula, em que NULL indica um caractere nulo (0x00), SPACE indica um espaço SBCS EBCDIC (0x40) e ZERO indica um zero decimal zonado não sinalizado (0xF0). Qualquer combinação correspondente dos bytes selecionados

em um registro de matriz estruturada fará com que o registro inteiro seja identificado como vazio. Se TRUNCATE-NULL-ARRAY-VALUES tem um valor definido, TRUNCATE-NULL-ARRAYS deve ser ativado.

XSDBIND = *value*

O nome completo da ligação XSD do z/OS UNIX. O DFHLS2SC criará o arquivo, mas não a estrutura de diretório, se ele ainda não existir. Se o parâmetro BUNDLE for especificado, exclua o caminho. A extensão do arquivo é .xsdbind.

DFHSC2LS: Esquema XML para Conversão de Linguagem de Alto Nível

O procedimento catalogado pelo DFHSC2LS gera uma estrutura de dados de linguagem de alto nível e uma ligação XML a partir de um esquema XML ou documento WSDL. Use o DFHSC2LS quando desejar criar um programa CICS que possa analisar ou criar XML. Este tópico lista as instruções de controle de tarefa, parâmetros simbólicos, parâmetros de entrada e suas descrições para DFHSC2LS.

Instruções de Controle da Tarefa para DFHSC2LS

JOB Inicia a tarefa.

EXEC Especifica o nome do procedimento (DFHSC2LS).

INPUT.SYSUT1 DD

Especifica a entrada. Os parâmetros de entrada geralmente são especificados no fluxo de entrada. No entanto, é possível defini-los em um conjunto de dados ou em um membro de um conjunto de dados particionados.

Parâmetros simbólicos

Os parâmetros simbólicos a seguir são definidos no DFHSC2LS:

JAVADIR = *path*

Especifica o nome do diretório Java que é usado pelo DFHSC2LS. O valor desse parâmetro é anexado a /usr/lpp/ fornecendo um nome de caminho completo de /usr/lpp/*path*.

Normalmente, você não especifica este parâmetro. O valor padrão é o valor que foi fornecido à tarefa de instalação do CICS (DFHISTAR) no parâmetro **JAVADIR**.

PATHPREF = *prefix*

Especifica um prefixo opcional que estende o caminho do diretório do z/OS UNIX usado em outros parâmetros. O padrão é a cadeia vazia.

Normalmente, você não especifica este parâmetro. O valor padrão é o valor que foi fornecido para a tarefa de instalação do CICS (DFHISTAR) no parâmetro **PATHPREF**.

TMPDIR = *tmpdir*

Especifica o local de um diretório no z/OS UNIX que o DFHSC2LS usa como uma área de trabalho provisória. O ID do usuário sob o qual a tarefa é executada deve ter permissão de leitura e gravação para esse diretório.

O valor padrão é /tmp.

TMPPFILE = *tmpprefix*

Especifica um prefixo que o DFHSC2LS usa para construir os nomes dos arquivos da área de trabalho provisória.

O valor padrão é SC2LS.

USSDIR = *path*

Especifica o nome do diretório do CICS TS no sistema de arquivos do z/OS UNIX. O valor desse parâmetro é anexado a /usr/lpp/cicsts/, fornecendo um nome de caminho completo de /usr/lpp/cicsts/*path*.

Normalmente, você não especifica este parâmetro. O valor padrão é o valor que foi fornecido à tarefa de instalação do CICS (DFHISTAR) no parâmetro **USSDIR**.

SERVICE = *value*

Use este parâmetro somente quando orientado a fazer isso pelo suporte IBM.

O Espaço de Trabalho Temporário

O DFHSC2LS cria os três arquivos temporários a seguir no tempo de execução:

```
tmpdir / tmpprefix .in  
tmpdir / tmpprefix .out  
tmpdir / tmpprefix .err
```

em que:

tmpdir é o valor especificado no parâmetro **TMPDIR**.

tmpprefix é o valor especificado no parâmetro **TMPFILE**.

Os nomes padrão para os arquivos (quando **TMPDIR** e **TMPFILE** não são especificados) são conforme a seguir:

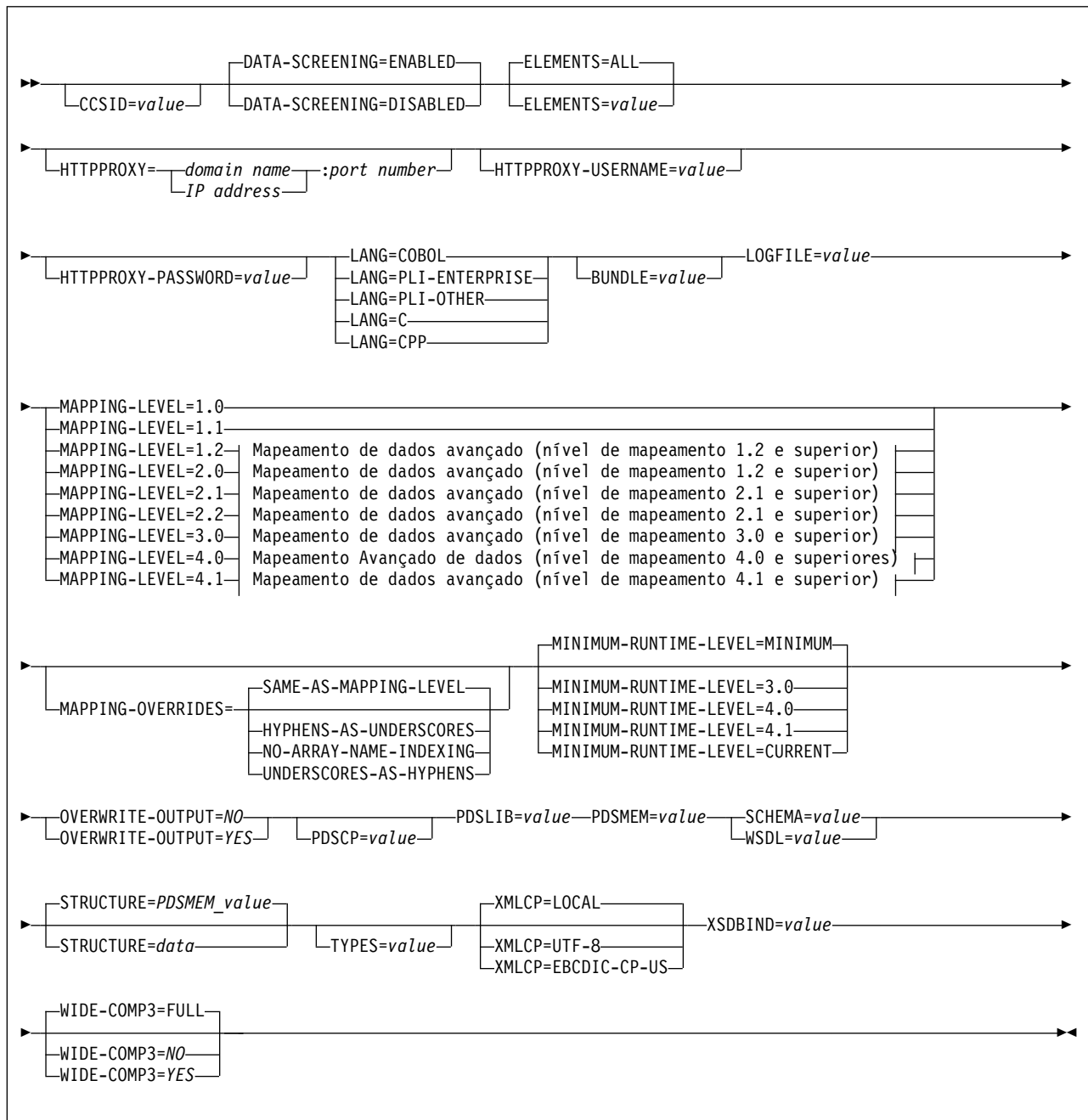
```
/tmp/SC2LS.in  
/tmp/SC2LS.out  
/tmp/SC2LS.err
```

Importante: O DFHSC2LS não bloqueia o acesso aos arquivos do z/OS UNIX ou aos membros do conjunto de dados. Portanto, se duas ou mais instâncias de DFHSC2LS forem executadas simultaneamente e usarem os mesmos arquivos da área de trabalho provisória, nada impede que uma tarefa sobrescreva os arquivos da área de trabalho enquanto outra tarefa está usando-os, levando a falhas imprevisíveis.

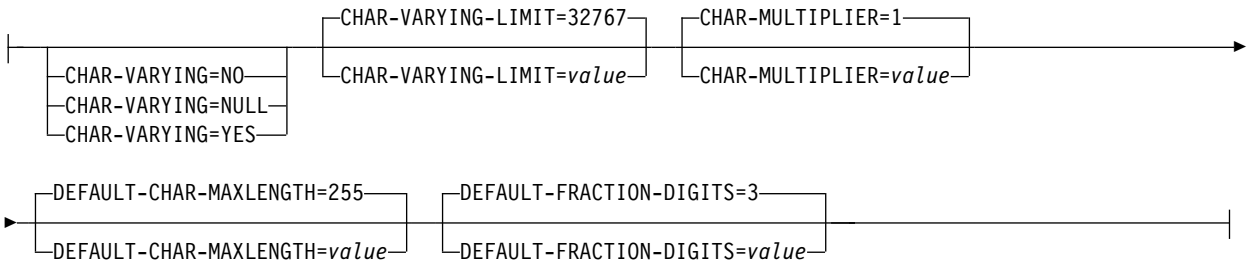
Portanto, você será avisado para planejar uma convenção de nomenclatura e procedimentos operacionais que evitem esta situação; por exemplo, é possível usar o parâmetro simbólico do sistema **SYSUID** para gerar nomes de arquivos da área de trabalho que sejam exclusivos para um usuário individual.

Estes arquivos temporários são excluídos antes do encerramento da tarefa.

Parâmetros de Entrada para DFHSC2LS



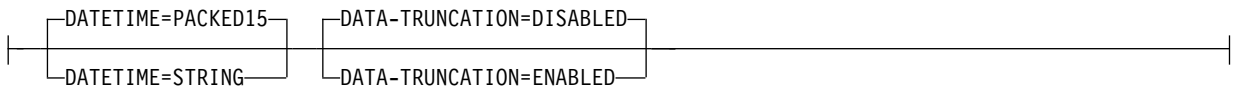
Mapeamento de Dados Avançado (Nível de Mapeamento 1.2 e Superior):



Mapeamento de Dados Avançado (Nível de Mapeamento 2.1 e Superior):



Mapeamento de Dados Avançado (Nível de Mapeamento 3.0):



Utilização de Parâmetros

- Você pode especificar os parâmetros de entrada em qualquer ordem.
- Cada parâmetro deve ser iniciado em uma nova linha.
- Um parâmetro (e seu caractere de continuação, se você usar um) não deve estender além da coluna 72; as colunas 73 a 80 devem conter espaços em branco.
- Se um parâmetro é muito longo para se ajustar em uma única linha, use um caractere de asterisco (*) no final da linha para indicar que o parâmetro continua na próxima linha. Tudo (incluindo espaços) antes do asterisco é considerado parte do parâmetro. Por exemplo:

```
XSDBIND=xsdbinddir*  
/app1
```

é equivalente a

```
XSDBIND=xsdbinddir/app1
```

- Um caractere # na primeira posição de caractere da linha é um caractere de comentário. A linha é ignorada.

Descrições de Parâmetros

CCSID = *value*

Especifica o CCSID que é usado no tempo de execução para codificar dados de caractere na estrutura de dados do aplicativo. O valor desse parâmetro substitui o valor do parâmetro de inicialização do sistema **LOCALCCSID**. O *value* deve ser um CCSID EBCDIC que seja suportado pelos serviços de conversão Java e z/OS (consulte z/OS Unicode Services User's Guide and Reference). Se

you do not specify this parameter, the data structure of the application will be encoded using the CCSID specified in the initialization parameter of the system.

This parameter can be used with any mapping level.

CHAR-MULTIPLIER = { 1 | *value* }

Specifies the number of bytes to be permitted for each character when the mapping level is 1.2 or more recent. The *value* of this parameter can be a positive integer in the range of 1 to 2,147,483,647. All mappings based on non-numeric characters are subject to this multiplier. Binary, numeric, zoned, and decimal compacted fields are not subject to this multiplier.

This parameter can be useful if, for example, you are planning to use DBCS characters where you can opt for a multiplier of 3 to allow space for possible shift-out and shift-in in each character of byte double in execution time.

When you configure **CCSID=1200** (indicating UTF-16), the only valid values for **CHAR-MULTIPLIER** are 2 or 4. When using UTF-16, the default value is 2. Use **CHAR-MULTIPLIER=2** when you expect the data of the application to contain characters that require 1 unit of UTF-16 encoding. Use **CHAR-MULTIPLIER=4** when you expect the data of the application to contain characters that require 2 units of UTF-16 encoding.

Nota: The configuration of **CHAR-MULTIPLIER** as 1 does not prevent the use of DBCS characters and its configuration as 2 does not prevent the use of UTF-16 surrogate pairs. However, if long characters are used routinely, some valid values may not fit in the allocated field. If a value for **CHAR-MULTIPLIER** greater than 1 is used, it may be possible to store more characters in the allocated field than are valid in XML. Care should be taken with regard to conformity with the range restrictions.

CHAR-VARYING = { **NO** | **NULL** | **YES** }

Specifies how data of variable-length characters are mapped when the mapping level is 1.2 or higher. Binary data of variable length are always mapped to a container or a variable structure. If you do not specify this parameter, the mapping will depend on the language specified. You can select from the following options:

NO The data of variable-length characters are mapped as fixed-length sequences.

NULL The data of variable-length characters are mapped as sequences with a null terminator.

YES The data of variable-length characters are mapped as a CHAR VARYING type in PL/I. In COBOL, C, and C++, the data of variable-length characters are mapped as a representation equivalent that consists of two related elements: the length of the data and the data.

CHAR-VARYING-LIMIT = { 32767 | *value* }

Specifies the maximum size of binary data and data of variable-length characters that are mapped to a language structure when the mapping level is 1.2 or higher. If the data of characters or binaries are larger than the value specified in this parameter, they will be

mapeados para um contêiner e o nome do contêiner será usado na estrutura de linguagem gerada. O *valor* pode variar de 0 ao padrão de 32 767 bytes.

DATA-SCREENING = { ENABLED | DISABLED }

Especifica se os dados fornecidos pelo aplicativo foram verificados com relação aos erros.

ENABLED

Quaisquer dados inconsistentes de tempo de execução fornecidos pelo aplicativo com a estrutura de linguagem são tratados como um erro e a mensagem DFHPI1010 é emitida. Uma resposta de erro é retornada ao aplicativo.

DISABLED

Os valores dos dados de tempo de execução fornecidos pelo aplicativo que são inconsistentes com a estrutura de linguagem são substituídos pelos valores padrão. Por exemplo, um zero substitui um valor inválido em um campo numérico. A mensagem DFHPI1010 não é emitida e uma resposta normal é retornada ao aplicativo. Esse recurso pode ser usado para evitar as respostas de erro INVALID_PACKED_DEC e INVALID_ZONED_DEC que são geradas a partir de campos de saída não inicializados.

DATA-TRUNCATION = { DISABLED | ENABLED }

Especifica se os dados de comprimento variável são tolerados em uma estrutura de campo de comprimento fixo:

DISABLED

Se os dados forem menores do que o comprimento fixo que o CICS está esperando, o CICS rejeitará os dados truncados e emitirá uma mensagem de erro.

ENABLED

Se os dados forem menores do que o comprimento fixo que o CICS está esperando, o CICS tolerará os dados truncados e processará os dados ausentes como valores nulos.

DATETIME = { PACKED15 | STRING }

Especifica que os campos xsd:dateTime são mapeados para o formato de dados CICS ABSTIME ou para texto:

PACKED15

Os campos xsd:dateTime são mapeados para o formato CICS ABSTIME.

CADEIA

Os campos xsd:dateTime são mapeados para texto. Esse mapeamento é o mesmo que todos os níveis de mapeamentos anteriores.

Você pode usar este parâmetro no nível de mapeamento 3.0.

DEFAULT-CHAR-MAXLENGTH = { 255 | *value* }

Especifica o comprimento da matriz padrão de dados de caractere em caracteres para mapeamentos, em que nenhum comprimento está implícito no documento de esquema XML ou documento WSDL, quando o nível de mapeamento é 1.2 ou superior. O *value* deste parâmetro pode ser um número inteiro positivo no intervalo de 1 a 2147483647.

DEFAULT-FRACTION-DIGITS = 3 | *value*

Especifica o número padrão de dígitos fracionários para uso em um tipo de

esquema decimal XML. O padrão é 3. Para COBOL, o intervalo válido é de 0 a 17 ou de 0 a 30 se o parâmetro **WIDE-COMP3** está sendo usado. Para C ou PL/I, o intervalo válido é 0 a 30.

ELEMENTS = { ALL | *value* }

Define uma lista de nomes locais de elementos globais a serem ativados. O valor padrão de ALL indica que todos os elementos globais estão ativados.

HTTPPROXY = { *domain name* | *IP address* } : *port number*

Se seu esquema XML ou documento WSDL contiver referências a outro esquema XML ou arquivos WSDL que estão localizados na Internet e o sistema no qual você está executando o DFHSC2LS usar um servidor proxy para acessar a Internet, especifique o nome de domínio ou endereço IP e o número da porta do servidor proxy. Por exemplo:

HTTPPROXY=proxy.example.com:8080

Em outros casos, este parâmetro não é necessário.

HTTPPROXY-USERNAME = *value*

Especifica o nome de usuário do proxy HTTP a ser usado com **HTTPPROXY-PASSWORD** se o sistema no qual você está executando o DFHSC2LS usar um servidor proxy HTTP para acessar a Internet e o servidor proxy HTTP usar autenticação básica. É possível usar esse parâmetro somente quando você também especifica **HTTPPROXY**.

HTTPPROXY-PASSWORD = *value*

Especifica a senha do proxy HTTP a ser usada com **HTTPPROXY-USERNAME** se o sistema no qual você está executando o DFHSC2LS usar um servidor proxy HTTP para acessar a Internet e o servidor proxy HTTP usar a autenticação básica. É possível usar esse parâmetro somente quando você também especifica **HTTPPROXY**.

INLINE-MAXOCCURS-LIMIT = { 1 | *value* }

Especifica se o conteúdo de repetição de variável sequencial é usado com base no atributo maxOccurs do atributo XML.

O parâmetro **INLINE-MAXOCCURS-LIMIT** está disponível somente no nível de mapeamento 2.1 em diante. O *value* de **INLINE-MAXOCCURS-LIMIT** pode ser um número inteiro positivo no intervalo de 0 a 32.767. Um valor 0 indica que o mapeamento sequencial não é usado. Um valor 1 garante que elementos opcionais sejam mapeados de forma sequencial. Se o *value* do atributo maxOccurs for maior que o *value* de **INLINE-MAXOCCURS-LIMIT**, o mapeamento baseado em contêiner será usado; caso contrário, o mapeamento sequencial será usado.

Ao decidir se deseja que listas de repetição variável sejam mapeadas sequencialmente, considere o comprimento de um único item de dados recorrentes. Se você tiver poucas instâncias de comprimento longo, o mapeamento baseado em contêiner será preferível; se você tiver muitas instâncias de comprimento curto, o mapeamento sequencial provavelmente será preferível.

LANG = **COBOL**

Especifica que a linguagem de programação da estrutura de linguagem de alto nível é COBOL.

LANG = **PLI-ENTERPRISE**

Especifica que a linguagem de programação da estrutura de linguagem de alto nível é Enterprise PL/I.

LANG = PLI-OTHER

Especifica que a linguagem de programação da estrutura de linguagem de alto nível é um nível de PL/I diferente de Enterprise PL/I.

LANG = C

Especifica que a linguagem de programação da estrutura de linguagem de alto nível é C.

LANG = CPP

Especifica que a linguagem de programação da estrutura de linguagem de alto nível é C++.

LOGFILE = value

O nome completo do arquivo z/OS UNIX no qual o DFHSC2LS grava seu log de atividades e informações de rastreamento. O DFHSC2LS criará o arquivo, mas não a estrutura de diretório, se ele ainda não existir.

Geralmente, você não usa esse arquivo, mas ele pode ser solicitado pela organização de serviço da IBM caso encontre problemas com o DFHSC2LS.

MAPPING-LEVEL = { 1.0 | 1.1 | 1.2 | 2.0 | 2.1 | 2.2 | 3.0 | 4.0 | 4.1 }

Especifica o nível de mapeamento para o assistente usar ao gerar a ligação XML e as estruturas de linguagem. É recomendável usar o nível de mapeamento mais recente disponível; para DFHSC2LS, é recomendável usar um nível de mapeamento igual a 3.0 ou superior.

3.0 O tipo de dados xsd:dateTime é mapeado para o formato CICS ASKTIME.

4.0 Use este nível de mapeamento com uma região do CICS TS 5.2 quando você desejar usar UTF-16.

4.1 Para suporte de matriz truncável, use este nível de mapeamento com uma região do CICS TS 5.2 que tenha o APAR PI67641 aplicado ou qualquer região do CICS TS 5.3 ou mais recente.

MAPPING-OVERRIDES = { SAME-AS-MAPPING-LEVEL | HYPHENS-AS-UNDERScores | NO-ARRAY-NAME-INDEXING | UNDERScores-AS-HYPHENS }

Especifica se o comportamento de mapeamento padrão é substituído pelo nível de mapeamento especificado. O valor padrão é SAME-AS-MAPPING-LEVEL.

SAME-AS-MAPPING-LEVEL

Este parâmetro gera estruturas de linguagem no mesmo estilo que o nível de mapeamento. Este é o padrão.

HYPHENS-AS-UNDERScores

Para PL/I somente. Este parâmetro converte quaisquer hífens no documento WSDL em sublinhados, em vez do caractere X, para melhorar a capacidade de leitura das estruturas de linguagem PL/I geradas. Para obter mais informações, consulte Esquema XML para Mapeamento de PL/I.

NO-ARRAY-NAME-INDEXING

Apenas para Enterprise PL/I. Assegura que os nomes de campos dentro de uma matriz sejam exclusivos somente dentro do escopo da estrutura de nível superior.

UNDERScores-AS-HYPHENS

Esta opção é ativada automaticamente no nível de mapeamento 4.0.

Apenas para COBOL. Converte quaisquer sublinhados no documento XML em hífens em vez do caractere X. Essa opção melhora a capacidade de leitura das estruturas de linguagem COBOL geradas. Se

ocorrerem quaisquer conflitos de nomes de campo, os campos serão numerados para garantir que sejam exclusivos. Para obter mais informações, consulte “Esquema XML para mapeamento COBOL” na página 320.

MINIMUM-RUNTIME-LEVEL = { MINIMUM | 3.0 | 4.1 CURRENT }

Especifica o ambiente de tempo de execução mínimo do CICS no qual a ligação XML pode ser implementada. Se você selecionar um nível que não corresponde aos outros parâmetros que você especificou, você receberá uma mensagem de erro. As opções que podem ser selecionadas são as seguintes:

MINIMUM

O menor nível de tempo de execução possível do CICS é alocado automaticamente com base nos parâmetros que você especificou.

3,0 Especifique o nível de tempo de execução 3.0 ou acima se deseja usar o assistente CICS XML e tirar vantagem dos mapeamentos de dados avançados.

4.0 O arquivo de ligação de serviço da web gerado é implementado com sucesso em um CICS TS 5.2 ou em uma região posterior. Com este nível de tempo de execução, é possível usar um nível de mapeamento 4.0 ou anterior para o parâmetro **MAPPING-LEVEL**. É possível utilizar qualquer parâmetro opcional neste nível.

4.1 O arquivo de ligação de serviços da web gerado é implementado com sucesso em uma região CICS V5.2 que possui o APAR PI67641 aplicado ou em qualquer região CICS V5.3 ou mais recente. Com este nível de tempo de execução, é possível utilizar um nível de mapeamento 4.1 ou anterior para o parâmetro **MAPPING-LEVEL**.

CURRENT

Use este nível de tempo de execução para implementar a ligação XML gerada em uma região CICS que possui o mesmo ambiente de tempo de execução que a região usada para gerar a ligação XML.

OVERWRITE-OUTPUT = NO | YES

Controla se **BUNDLES** existentes do CICS no sistema de arquivos podem ser sobrescritos.

NO Qualquer **BUNDLE** existente não é substituído. Se um **BUNDLE** existente for localizado, o DFHLS2SC emitirá a mensagem de erro DFHPI9689E e finalizará.

YES Qualquer **BUNDLE** existente é substituído. Se um **BUNDLE** existente for localizado, o DFHLS2SC emitirá a mensagem DFHPI9683W para informá-lo que o arquivo foi substituído.

PDSCP = value

Especifica a página de códigos que é usada nos membros do conjunto de dados particionados, em que *value* é um número de CCSID ou um número da página de códigos Java. Se você não especificar esse parâmetro, a página de códigos do z/OS UNIX System Services será usada. Por exemplo, você pode especificar PDSCP=037.

PDSLIB = value

Especifica o nome do conjunto de dados particionados que contém a linguagem de alto nível gerada.

PDSMEM = *value*

Especifica o prefixo de 1 a 6 caracteres que o DFHSC2LS usa para gerar o nome do membro do conjunto de dados particionados que conterá as estruturas de linguagem de alto nível.

O DFHSC2LS gera um membro do conjunto de dados particionados para cada operação. Ele gera o nome do membro anexando um número de 2 dígitos no prefixo.

SCHEMA = *value*

O nome completo do arquivo z/OS UNIX a partir do qual o esquema XML é lido. O DFHSC2LS criará o arquivo, mas não a estrutura de diretório, se ele ainda não existir.

É possível usar um esquema XML ou um documento WSDL como entrada para DFHSC2LS. Deve-se especificar esse parâmetro ou o parâmetro **WSDL**, mas não ambos, para indicar a origem da entrada.

STRUCTURE = { *PDSMEM_value* | *data* }

O nome da estrutura de dados de nível superior em C e C++. O valor padrão é o valor do parâmetro **PDSMEM**.

TYPES = *value*

Define uma lista de nomes locais de tipo global para ativar. Um *value* igual a ALL indica que todos os tipos globais estão ativados. Por padrão, os tipos globais não são ativados.

WIDE-COMP3 = { **FULL** | **NO** | **YES** }

Controla o tamanho máximo do comprimento variável decimal compactado na estrutura de linguagem COBOL ou PL/I gerada.

FULL Para COBOL e PL/I. DFHJS2LS gera um campo decimal compactado que é grande o suficiente para conter todos os valores válidos. O tamanho máximo são 31 dígitos. Este é o padrão.

NO Apenas para COBOL. DFHJS2LS limita o comprimento variável decimal compactado a 18 ao gerar a estrutura de linguagem COBOL tipo COMP-3. Se o tamanho decimal compactado for maior que 18, a mensagem DFHPI9022W será emitida para indicar que o tipo especificado está sendo limitado a um total de 18 dígitos.

YES Apenas para COBOL. DFHJS2LS suporta o tamanho máximo de 31 ao gerar a estrutura de linguagem COBOL tipo COMP-3.

Nota: As opções NO e YES geram campos que são incapazes de representar todos os valores válidos; a opção FULL evita esse problema. No entanto, a opção FULL permite que alguns valores inválidos sejam representados no campo decimal compactado. Por exemplo, se um esquema indicar que há um máximo de cinco dígitos e um máximo de dois dígitos fracionários, a opção FULL gerará um campo decimal compactado que permite sete dígitos e isso permite espaço para valores válidos, como 25000 e 999,99, mas também fornece espaço para alguns valores inválidos como 9999,99. Quando você usar a opção FULL, tome cuidado para não gerar valores inválidos em dados do aplicativo.

WSDL = *value*

O nome completo do documento WSDL do z/OS UNIX.

É possível usar um esquema XML ou um documento WSDL como entrada para DFHSC2LS. Deve-se especificar esse parâmetro ou o parâmetro **SCHEMA**, mas não ambos, para indicar a origem da entrada.

XMLCP = { **LOCAL** | **UTF-8** | **EBCDIC-CP-US** }

Especifica a página de códigos que é utilizada para gerar a ligação XML.

LOCAL

Este valor é o padrão. Ele especifica que o XML é gerado usando a página de códigos local e nenhuma tag de codificação é gerada no esquema XML.

UTF-8 Especifica que o XML é gerado usando a página de códigos UTF-8. Uma tag de codificação é gerada no esquema XML. Se você especificar esta opção, deverá assegurar que a codificação permaneça correta ao copiar o esquema XML entre diferentes plataformas.

EBCDIC-CP-US

Especifica que o XML é gerado usando a página de códigos US EBCDIC. Uma tag de codificação é gerada no esquema XML.

XSDBIND = *value*

O nome completo da ligação XML do z/OS UNIX. O DFHSC2LS criará o arquivo, mas não a estrutura de diretório, se ele ainda não existir. A extensão do arquivo é .xsdbind.

Linguagem de Alto Nível e Mapeamento de Esquema XML

Utilize os assistentes do CICS para gerar mapeamentos entre as estruturas de linguagem de alto nível e os esquemas XML ou documentos WSDL. Os assistentes do CICS também geram esquemas XML ou documentos WSDL a partir de estruturas de dados de linguagem de alto nível ou vice-versa.

Os programas utilitários DFHSC2LS e DFHLS2SC são conhecidos coletivamente como o assistente XML do CICS. Os programas utilitários DFHWS2LS e DFHLS2WS são conhecidos coletivamente como o assistente de serviços da web do CICS.

- DFHLS2SC e DFHLS2WS mapeiam estruturas de linguagem de alto nível para esquemas XML e documentos WSDL respectivamente.
- DFHSC2LS e DFHWS2LS mapeiam esquemas XML e documentos WSDL para estruturas de linguagem de alto nível.

Os dois mapeamentos não são simétricos:

- Se você processar uma estrutura de dados de linguagem com DFHLS2SC ou DFHLS2WS e, em seguida, processar o esquema XML ou documento WSDL resultante com o programa utilitário complementar (respectivamente DFHSC2LS ou DFHWS2LS), não espere que a estrutura de dados final seja igual à original. No entanto, a estrutura de dados final é logicamente equivalente à original.
- Se você processar um esquema XML ou documento WSDL com DFHSC2LS ou DFHWS2LS e, em seguida, processar a estrutura de linguagem resultante com o programa utilitário complementar (respectivamente DFHLS2SC ou DFHLS2WS), não espere que o esquema XML no esquema XML ou documento WSDL final seja igual ao original.
- Em alguns casos, DFHSC2LS e DFHWS2LS geram estruturas de linguagem que não são suportadas pelo DFHLS2SC e DFHLS2WS.

Deve-se codificar estruturas de linguagem processadas por DFHLS2SC e DFHLS2WS de acordo com as regras da linguagem, conforme implementado nos compiladores de linguagem que o CICS suporta.

Limitações de mapeamento de dados ao usar os assistentes do CICS:

O CICS suporta mapeamentos de dados bidirecionais entre as estruturas de linguagem de alto nível e os esquemas XML ou documentos WSDL que estão em conformidade com o WSDL versão 1.1 ou 2.0, com determinadas limitações. Essas limitações se aplicam somente às ferramentas DFHWS2LS e DFHSC2LS e variam de acordo com o nível de mapeamento.

Limitações em Todos os Níveis de Mapeamento

- Somente ligações SOAP que usam codificação literal são suportadas. Portanto, deve-se configurar o atributo `use` com um valor literal; `use="encoded"` não é suportado.
- Definições de tipo de dados devem ser codificadas usando a linguagem XML Schema Definition (XSD). No esquema, tipos de dados usados na mensagem SOAP devem ser declarados explicitamente.
- O comprimento de algumas palavras-chave na descrição de serviços da web é limitado. Por exemplo, operação, ligação e nomes de partes são limitados a 255 caracteres. Em alguns casos, o comprimento máximo do nome da operação pode ser um pouco menor.
- Quaisquer falhas de SOAP definidas na descrição do serviço da web são ignoradas. Se você deseja que um aplicativo do provedor de serviços envie uma mensagem de falha de SOAP, use o comando **EXEC CICS SOAPFAULT**.
- DFHWS2LS e DFHSC2LS suportam somente um único elemento `<xsd:any>` em um escopo específico. Por exemplo, o fragmento de esquema a seguir não é suportado:

```
<xsd:sequence>
<xsd:any/>
<xsd:any/>
</xsd:sequence>
```

Aqui, `<xsd:any>` pode especificar `minOccurs` e `maxOccurs` se necessário. Por exemplo, o fragmento de esquema a seguir é suportado:

```
<xsd:sequence>
<xsd:any minOccurs="2" maxOccurs="2"/>
</xsd:sequence>
```

- Referências cíclicas não são suportadas. Por exemplo, onde tipo A contém tipo B que, por sua vez, contém tipo A.
- A recorrência não é suportada nos elementos de grupo, tais como os elementos `<xsd:choice>`, `<xsd:sequence>`, `<xsd:group>` ou `<xsd:all>`. Por exemplo, o fragmento de esquema a seguir não é suportado:

```
<xsd:choice maxOccurs="2">
<xsd:element name="name1" type="string"/>
</xsd:choice>
```

A exceção é no nível de mapeamento 2.1 e superior, em que `maxOccurs="1"` e `minOccurs="0"` são suportados nestes elementos.

- DFHSC2LS e DFHWS2LS não suportam tipos de dados e elementos na mensagem SOAP que são derivados dos tipos de dados e elementos declarados no esquema XML a partir do atributo `xi:type` ou de um grupo de substituição, exceto no nível de mapeamento 2.2 e superior se o elemento pai ou tipo é definido como abstrato.

- Elementos `<xsd:sequence>` e `<xsd:group>` integrados dentro de um elemento `<xsd:choice>` não são suportados antes do nível de mapeamento 2.2. Elementos `<xsd:choice>` e `<xsd:all>` integrados dentro de um elemento `<xsd:choice>` nunca são suportados.

Suporte Aprimorado no Nível de Mapeamento 1.1 e Superior

Quando o nível de mapeamento é 1.1 ou superior, DFHWS2LS fornece suporte para os elementos XML e tipo de elemento a seguir:

- O elemento `<xsd:list>`.
- O elemento `<xsd:union>`.
- O tipo `xsd:anySimpleType`.
- O elemento `<xsd:attribute>`. No nível de mapeamento 1.0, este elemento é ignorado.

Suporte Aprimorado no Nível de Mapeamento 2.1 e Superior

Quando o nível de mapeamento é 2.1 ou superior, o DFHWS2LS suporta os seguintes elementos XML e atributos do elemento:

- O elemento `<xsd:any>`.
- O tipo `xsd:anyType`.
- Elementos abstratos. Em níveis anteriores de mapeamento, elementos abstratos são suportados somente como tipos não terminal em uma hierarquia de herança.
- Os atributos `maxOccurs` e `minOccurs` nos elementos `<xsd:all>`, `<xsd:choice>`, e `<xsd:sequence>`, somente quando `maxOccurs="1"` e `minOccurs="0"`.
- Campos "FILLER" em campos COBOL e "*" no PL/I são suprimidos. Os campos não aparecem no WSDL gerado e um intervalo apropriado é deixado nas estruturas de dados no tempo de execução.

Suporte Aprimorado no Nível de Mapeamento 2.2 e Superior

Quando o nível de mapeamento é 2.2 ou superior, DFHSC2LS e DFHWS2LS fornecem suporte melhorado para o elemento `<xsd:choice>`, que suporta um máximo de 255 opções no elemento `<xsd:choice>`. Para obter mais informações sobre o suporte `<xsd:choice>`, consulte "Suporte para `<xsd:choice>`" na página 356.

No nível de mapeamento 2.2 e superior, os assistentes do CICS suportam os mapeamentos XML a seguir:

- Grupos de Substituição
- Valores fixos para elementos
- Tipos de dados abstratos

Elementos `<xsd:sequence>` e `<xsd:group>` integrados dentro de um elemento `<xsd:choice>` são suportados no nível de mapeamento 2.2 e superior. Por exemplo, o fragmento de esquema a seguir é suportado:

```
<xsd:choice>
<xsd:element name ="name1" type="string"/>
<xsd:sequence/>
</xsd:choice>
```

Se o elemento-pai ou tipo na mensagem SOAP é definido como abstrato, DFHSC2LS e DFHWS2LS suportam tipos de dados e elementos que são derivados dos tipos de dados e elementos declarados no esquema XML.

Suporte Aprimorado no Nível de Mapeamento 3.0 e Superior

Quando o nível de mapeamento é 3.0 ou superior, os assistentes do CICS suportam os aprimoramentos de mapeamento a seguir:

- DFHSC2LS e DFHWS2LS mapeiam tipos de dados `xsd:dateTime` para o formato CICS ASKTIME.
- O DFHLS2WS pode gerar um documento WSDL e a ligação de serviço da web a partir de um aplicativo que usa muitos contêineres em vez de somente um contêiner.
- Tolerando dados truncados que são descritos por uma estrutura de dados de comprimento fixo. É possível configurar este comportamento usando o parâmetro **DATA-TRUNCATION** nos assistentes do CICS.

Suporte aprimorado no nível de mapeamento 4.0 e superior

Quando o nível de mapeamento é 4.0 ou superior, os assistentes do CICS suportam os aprimoramentos de mapeamento a seguir:

No nível de mapeamento 4.0 e superior, DFHLS2SC e DFHLS2WS suportam a cláusula COBOL OCCURS DEPENDING ON e suportam o mapeamento de matrizes de caracteres COBOL para sequências XML. É possível configurar este comportamento usando o parâmetro **CHAR-OCCURS** nos assistentes do CICS.

- Deve-se especificar o parâmetro **DATA-TRUNCATION=ENABLED**.
- OCCURS DEPENDING ON complexo não é suportado. Esta limitação significa que OCCURS DEPENDING ON é suportado somente para o último campo de uma estrutura.
- O CICS não suporta nomes qualificados (usando a palavra-chave 'OF') como o destino de uma cláusula OCCURS DEPENDING ON, por exemplo FIELD1 OF STRUCTURE1.
- O CICS não suporta a palavra-chave UNBOUNDED. Deve-se especificar o tamanho máximo da tabela que é esperado pelo aplicativo.

No nível de mapeamento 4.0 e superior, os serviços da web do CICS suportam a conversão de dados do aplicativo que são codificados usando UTF-16 Unicode.

- Quando você usa DFHLS2WS ou DFHLS2SC, é possível ativar este comportamento usando tipos de dados específicos de linguagem para UTF-16.
- Quando você usa DFHWS2LS ou DFHSC2LS, é possível ativar este comportamento configurando CCSID=1200.
- CICS suporta somente uma única página de códigos Unicode, " UTF-16BE com IBM Private Use Area " (CCSID 1200).
- A conversão de dados do aplicativo que são codificados usando UTF-8 não é suportada.

Nota: DFHLS2WS e DFHLS2SC não suportam a cláusula COBOL GROUP USAGE NATIONAL.

Mapeamento de COBOL para Esquema XML:

Os programas utilitários DFHLS2SC e DFHLS2WS suportam mapeamentos entre estruturas de dados COBOL e definições de esquema XML.

Os nomes COBOL são convertidos em nomes XML de acordo com as regras a seguir:

1. Nomes duplicados se tornam exclusivos pela adição de um ou mais dígitos numéricos.
Por exemplo, duas instâncias de year se tornam year e year1.
2. Hifens são substituídos por caracteres de sublinhado. Sequências de hifens contíguos são substituídas por sublinhados contíguos.
Por exemplo, current-user--id se torna current_user_id.
3. Segmentos de nomes que são delimitados por hifens e que contêm somente caracteres em maiúsculas são convertidos em minúsculas.
Por exemplo, CA-REQUEST-ID se torna ca_request_id.
4. Um caractere de sublinhado de orientação é incluído nos nomes que iniciam com um caractere numérico.
Por exemplo, 9A-REQUEST-ID se torna _9a_request_id.

O CICS mapeia elementos de descrição de dados COBOL para elementos de esquema de acordo com a tabela a seguir. Elementos de descrição de dados COBOL que não são mostrados na tabela não são suportados pelo DFHLS2SC ou DFHLS2WS. As seguintes restrições também se aplicam:

- Os itens de descrição de dados com os números de nível 66 e 77 não são suportados. Os itens de descrição de dados com um número de nível 88 são ignorados.
- As seguintes cláusulas nas entradas de descrição de dados não são suportadas:
REDEFINES
RENAMES; que é o nível 66
DATE FORMAT
- As seguintes cláusulas nos itens de descrição de dados são ignoradas:
BLANK WHEN ZERO
JUSTIFIED
VALUE
- A cláusula SIGN, SIGN TRAILING, é suportada. A cláusula SIGN, SIGN LEADING, é suportada somente quando o nível de mapeamento especificado em DFHLS2SC ou DFHLS2WS é 1.2 ou superior.
- SEPARATE CHARACTER é suportado em um nível de mapeamento de 1.2 ou superior para ambas as cláusulas SIGN TRAILING e SIGN LEADING.
- As seguintes frases na cláusula USO não são suportadas:
OBJECT REFERENCE
POINTER
FUNCTION-POINTER
PROCEDURE-POINTER
- As frases a seguir na cláusula USAGE são suportadas em um nível de mapeamento de 1.2 ou superior:
COMPUTATIONAL-1
COMPUTATIONAL-2
- Os únicos caracteres de PICTURE suportados para os itens de descrição de dados DISPLAY e COMPUTATIONAL-5 são 9, S e Z.
- Os caracteres de PICTURE que são suportados para os itens de descrição de dados PACKED-DECIMAL são 9, S, V e Z.
- Os únicos caracteres de PICTURE que são suportados para os itens de descrição de dados numéricos editados são 9 e Z.

- Se o parâmetro **MAPPING-LEVEL** for configurado como 1.2 ou superior e o parâmetro **CHAR-VARYING** for configurado como NULL, as matrizes de caracteres serão mapeadas para um `xsd:string` e serão processadas como sequências com terminação nula.
- Se o parâmetro **MAPPING-LEVEL** for configurado como 1.2 ou superior e o parâmetro **CHAR-VARYING** for configurado como BINARY, as matrizes de caracteres serão mapeadas para `xsd:base64Binary` e serão processadas como dados binários.
- Se o parâmetro **MAPPING-LEVEL** for configurado como 1.2 ou superior e o parâmetro **CHAR-VARYING** for configurado como COLLAPSE, espaços em branco finais serão ignorados para sequências.
- A cláusula OCCURS DEPENDING ON é suportada em um nível de mapeamento de 4.0 ou superior. OCCURS DEPENDING ON complexo não é suportado. Isso significa que OCCURS DEPENDING ON é suportado somente para o último campo de uma estrutura.
- A cláusula OCCURS INDEXED BY é suportada em qualquer nível de mapeamento.
- A cláusula OCCURS é suportada para até 65535 VEZES de ocorrências. Isso significa que OCCURS *n* TIMES em que *n* é maior do que 65535 não é suportado.

Descrição de Dados COBOL	simpleType de Esquema
<div>PIC X(<div><i>n</i></div>)</div> <div>PIC A(<div><i>n</i></div>)</div> <div>PIC G(<div><i>n</i></div>) DISPLAY-1</div> <div>PIC N(<div><i>n</i></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>em que <i>n</i> é o valor máximo que pode ser representado pelo padrão de '9' caracteres.</p>

Descrição de Dados COBOL	simpleType de Esquema
PIC S9(z) DISPLAY em que $5 \leq z \leq 9$	<pre><xsd:simpleType> <xsd:restriction base="xsd:int"> <xsd:minInclusive value="- n"/> <xsd:maxInclusive value=" n"/> </xsd:restriction> </xsd:simpleType></pre> <p>em que n é o valor máximo que pode ser representado pelo padrão de '9' caracteres.</p>
PIC S9(z) DISPLAY em que $9 < z$	<pre><xsd:simpleType> <xsd:restriction base="xsd:long"> <xsd:minInclusive value="- n"/> <xsd:maxInclusive value=" n"/> </xsd:restriction> </xsd:simpleType></pre> <p>em que n é o valor máximo que pode ser representado pelo padrão 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>em que n é o valor máximo que pode ser representado pelo padrão de '9' caracteres.</p>
PIC 9(z) DISPLAY em que $5 \leq z \leq 9$	<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedInt"> <xsd:minInclusive value="0"/> <xsd:maxInclusive value=" n"/> </xsd:restriction> </xsd:simpleType></pre> <p>em que n é o valor máximo que pode ser representado pelo padrão de '9' caracteres.</p>
PIC 9(z) DISPLAY em que $9 < z$	<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedLong"> <xsd:minInclusive value="0"/> <xsd:maxInclusive value=" n"/> </xsd:restriction> </xsd:simpleType></pre> <p>em que n é o valor máximo que pode ser representado pelo padrão de '9' caracteres.</p>

Descrição de Dados COBOL	simpleType de Esquema
PIC S9(n) COMP PIC S9(n) COMP-4 PIC S9(n) COMP-5 PIC S9(n) BINARY em que $n \leq 4$.	<pre> <xsd:simpleType> <xsd:restriction base="xsd:short"> </xsd:restriction> </xsd:simpleType> </pre>
PIC S9(n) COMP PIC S9(n) COMP-4 PIC S9(n) COMP-5 PIC S9(n) BINARY em que $5 \leq n \leq 9$.	<pre> <xsd:simpleType> <xsd:restriction base="xsd:int"> </xsd:restriction> </xsd:simpleType> </pre>
PIC S9(n) COMP PIC S9(n) COMP-4 PIC S9(n) COMP-5 PIC S9(n) BINARY em que $9 < n$.	<pre> <xsd:simpleType> <xsd:restriction base="xsd:long"> </xsd:restriction> </xsd:simpleType> </pre>

Descrição de Dados COBOL	simpleType de Esquema
PIC 9(n) COMP PIC 9(n) COMP-4 PIC 9(n) COMP-5 PIC 9(n) BINARY em que $n \leq 4$.	<pre> <xsd:simpleType> <xsd:restriction base="xsd:unsignedShort"> </xsd:restriction> </xsd:simpleType> </pre>
PIC 9(n) COMP PIC 9(n) COMP-4 PIC 9(n) COMP-5 PIC 9(n) BINARY em que $5 \leq n \leq 9$.	<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 em que $9 < n$.	<pre> <xsd:simpleType> <xsd:restriction base="xsd:unsignedLong"> </xsd:restriction> </xsd:simpleType> </pre>

Descrição de Dados COBOL	simpleType de Esquema
PIC S9(<i>m</i>)V9(<i>n</i>) COMP-3	<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>em que $p = m + n$.</p>
PIC 9(<i>m</i>)V9(<i>n</i>) COMP-3 PIC S9(<i>m</i>) COMP-3 Suportado no nível de mapeamento 3.0 quando DATETIME=PACKED15	<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>em que $p = m + n$.</p> <pre><xsd:simpleType> <xsd:restriction base="xsd:dateTime"> </xsd:restriction> </xsd:simpleType></pre> <p>O formato do registro de data e hora é CICS ABSTIME.</p>
PIC S9(<i>m</i>)V9(<i>n</i>) DISPLAY Suportado no nível de mapeamento 1.2 e superior	<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>em que $p = m + n$.</p>
COMP-1 Suportado no nível de mapeamento 1.2 e superior Nota: A representação de dados IBM Hexadecimal Floating Point (HFP) não é exatamente a mesma que a representação IEEE-754-1985 usada para XML. Alguns valores podem não ser convertidos exatamente de uma representação para a outra. Alguns valores extremamente grandes ou pequenos podem não ser válidos para tipos de dados xsd:float. Alguns valores podem perder a precisão quando convertidos para ou a partir da representação HFP. Se conversões precisas forem importantes, considere substituir o uso de tipos de dados COMP-1 por alternativas de precisão fixas.	<pre><xsd:simpleType> <xsd:restriction base="xsd:float"> </xsd:restriction> </xsd:simpleType></pre>

Descrição de Dados COBOL	simpleType de Esquema
<p>COMP-2</p> <p>Suportado no nível de mapeamento 1.2 e superior</p> <p>Nota: A representação de dados IBM Hexadecimal Floating Point (HFP) não é exatamente a mesma que a representação IEEE-754-1985 usada para XML. Alguns valores podem não ser convertidos exatamente de uma representação para a outra. Alguns valores extremamente grandes ou pequenos podem não ser válidos para os tipos de dados <code>xsd:double</code>. Alguns valores podem perder a precisão quando convertidos para ou a partir da representação HFP. Se conversões precisas forem importantes, considere substituir o uso de tipos de dados COMP-2 por alternativas de precisão fixas.</p>	<pre><xsd:simpleType> <xsd:restriction base="xsd:double"> </xsd:restriction> </xsd:simpleType></pre>
<p><i>data description</i> OCCURS <i>n</i> TIMES</p>	<pre><xsd:element name= "field-name" minOccurs= "n" maxOccurs= "n"> ... </xsd:element></pre> <p>O conteúdo do elemento depende do tipo de dados usado.</p>
<p><i>data description</i> OCCURS <i>n</i> TO <i>m</i> TIMES DEPENDENT ON <i>t</i></p> <p>Suportado no nível de mapeamento 4.0</p>	<pre><xsd:element name= "field-name" minOccurs= "n" maxOccurs= "m"> ... </xsd:element></pre>

Descrição de Dados COBOL	simpleType de Esquema
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>Quando 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>Ele é uma sequência.</p> <p>Quando 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>Isto é uma matriz de caracteres únicos.</p>

Descrição de Dados COBOL	simpleType de Esquema
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>	Quando 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 Quando 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> No tempo de execução, CICS preenche o campo de estrutura de dados do aplicativo com dados UTF-16.

Esquema XML para mapeamento COBOL:

Os programas utilitários DFHSC2LS e DFHWS2LS suportam mapeamentos entre definições de esquema XML e estruturas de dados COBOL.

Os assistentes do CICS geram nomes exclusivos e válidos para variáveis COBOL a partir dos nomes de elemento de esquema usando as regras a seguir:

1. Palavras reservadas COBOL são prefixadas com ' X' inicial.
Por exemplo, DISPLAY se torna XDISPLAY.
2. Caracteres diferentes de A-Z, a-z, 0-9 ou hífen são substituídos por ' X' inicial.
Por exemplo, monthly_total se torna monthlyXtotal . É possível usar o parâmetro **MAPPING-OVERRIDES** para mudar o modo como outros caracteres são

manipulados. Por exemplo, se você configurar o valor `UNDERSCORES-AS-HYPHENS`, qualquer sublinhado no XML será convertido em um hífen em vez de um X. Então, `monthly_total` se torna `monthly-total`.

3. Se o último caractere for um hífen, ele será substituído por 'X' inicial.

Por exemplo, `ca-request-` se torna `ca-requestX`.

4. Se o esquema especificar que a variável possui variação de cardinalidade (ou seja, `minOccurs` e `maxOccurs` são especificados em um `xsd:element` com valores diferentes) e o nome de elemento do esquema for maior do que 23 caracteres, ele será truncado para esse comprimento.

Se o esquema especificar que a variável possui cardinalidade fixa e o nome de elemento do esquema for maior do que 28 caracteres, ele será truncado para esse comprimento.

5. Nomes duplicados no mesmo escopo se tornam exclusivos pela adição de um ou dois dígitos numéricos na segunda instância e nas instâncias subsequentes do nome.

Por exemplo, três instâncias de `year` se tornam `year`, `year1` e `year2`.

6. Cinco caracteres são reservados para as sequências `-cont` ou `-num`, os quais são usados quando o esquema especifica que a variável possui variação de cardinalidade; ou seja, quando `minOccurs` e `maxOccurs` são especificados com valores diferentes.

Para obter mais informações, consulte “Matrizes de Elementos Variáveis” na página 347.

7. Para atributos, as regras anteriores são aplicadas ao nome de elemento. O prefixo `attr-` é incluído no nome de elemento e é seguido por `-value` ou `-exist`. Se o comprimento total for maior do que 28 caracteres, o nome de elemento será truncado. Para obter mais informações, consulte “Suporte para Atributos XML” na página 352.

O atributo anulável possui regras especiais. O prefixo `attr-` é incluído, mas `nil-` também é incluído no início do nome de elemento. O nome de elemento é seguido por `-value`. Se o comprimento total for maior do que 28 caracteres, o nome de elemento será truncado.

O comprimento total do nome resultante é de 30 caracteres ou menos.

DFHSC2LS e DFHWS2LS mapeiam tipos de esquema para elementos de descrição de dados COBOL usando o nível de mapeamento especificado de acordo com a tabela a seguir. Observe os pontos a seguir:

- Se o parâmetro **MAPPING-LEVEL** for configurado como 1.2 ou superior e o parâmetro **CHAR-VARYING** for configurado como NULL, os dados de caractere de comprimento variável serão mapeados para sequências com terminação nula e um caractere extra será alocado para o terminador NULL.
- Se o parâmetro **MAPPING-LEVEL** for configurado como 1.2 ou superior e o parâmetro **CHAR-VARYING** for configurado como YES, os dados de caractere de comprimento variável serão mapeados para dois elementos relacionados: um campo de comprimento e um campo de dados. Por exemplo:

```
<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"/>
```

mapeia para:

15 textString-length PIC S9999 COMP-5 SYNC
15 textString PIC X(10000)

Tipo Simples de Esquema	Descrição de Dados COBOL
<pre><xsd:simpleType> <xsd:restriction base="xsd:anyType"> </xsd:restriction> </xsd:simpleType></pre>	<p>Nível de mapeamento 2.0 e abaixo:</p> <p>Não-suportado</p> <p>Nível de Mapeamento 2.1:</p> <p>Suportado</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:anySimpletype"> </xsd:restriction> </xsd:simpleType></pre>	<p>Nível de mapeamento 1.0:</p> <p>Não-suportado</p> <p>Nível de mapeamento 1.1 e 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>em que <i>type</i> é um de:</p> <ul style="list-style-type: none"> string normalizedString token Name NMTOKEN language NCName ID IDREF ENTITY hexBinary 	<p>Todos os níveis de mapeamento:</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>em que <i>type</i> é um de:</p> <ul style="list-style-type: none"> string normalizedString token Name NMTOKEN language NCName ID IDREF ENTITY 	<p>Quando CCSID=1200 no nível de mapeamento 4.0 e superior:</p> <p>PIC N(</p> <p>z</p> <p>) USAGE NATIONAL</p>

Tipo Simples de Esquema	Descrição de Dados COBOL
<pre><xsd:simpleType> <xsd:restriction base="xsd: type" </xsd:restriction> </xsd:simpleType></pre> <p>em que <i>type</i> é um de:</p> <ul style="list-style-type: none"> duration date time gDay gMonth gYear gMonthDay gYearMonth 	<p>Todos os níveis de mapeamento:</p> <p>PIC X(32)</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:dateTime" </xsd:restriction> </xsd:simpleType></pre>	<p>Nível de mapeamento 1.2 e abaixo:</p> <p>PIC X(32)</p> <p>Nível de mapeamento 2.0 e superior:</p> <p>PIC X(40)</p> <p>Nível de mapeamento 3.0 e superior:</p> <p>PIC S9(15) COMP-3</p> <p>O formato é CICS ABSTIME.</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd: type"> </xsd:restriction> </xsd:simpleType></pre> <p>em que <i>type</i> é um de:</p> <ul style="list-style-type: none"> byte unsignedByte 	<p>Todos os níveis de mapeamento:</p> <p>PIC X DISPLAY</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:short"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos os níveis de mapeamento:</p> <p>PIC S9999 COMP-5 SYNC</p> <p>ou</p> <p>PIC S9999 DISPLAY</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedShort"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos os níveis de mapeamento:</p> <p>PIC 9999 COMP-5 SYNC</p> <p>ou</p> <p>PIC 9999 DISPLAY</p>

Tipo Simples de Esquema	Descrição de Dados COBOL
<pre><xsd:simpleType> <xsd:restriction base="xsd:integer"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos os níveis de mapeamento:</p> <p>PIC S9(18) COMP-3</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:int"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos os níveis de mapeamento:</p> <p>PIC S9(9) COMP-5 SYNC</p> <p>ou</p> <p>PIC S9(9) DISPLAY</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedInt"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos os níveis de mapeamento:</p> <p>PIC 9(9) COMP-5 SYNC</p> <p>ou</p> <p>PIC 9(9) DISPLAY</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:long"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos os níveis de mapeamento:</p> <p>PIC S9(18) COMP-5 SYNC</p> <p>ou</p> <p>PIC S9(18) DISPLAY</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedLong"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos os níveis de mapeamento:</p> <p>PIC 9(18) COMP-5 SYNC</p> <p>ou</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 os níveis de mapeamento:</p> <p>Se WIDE-COMP3=FULL:</p> <p>PIC 9(</p> <p>m</p> <p>)V9(</p> <p>n</p> <p>) COMP-3</p> <p>Caso contrário:</p> <p>PIC 9(</p> <p>p</p> <p>)V9(</p> <p>n</p> <p>) COMP-3</p> <p>em que $p = m - n$.</p>

Tipo Simples de Esquema	Descrição de Dados COBOL
<pre><xsd:simpleType> <xsd:restriction base="xsd:boolean"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos os níveis de mapeamento:</p> <p>PIC X DISPLAY</p> <p>O valor x'00' significa falso, x'01' significa verdadeiro.</p>
<pre><xsd:simpleType> <xsd:list> <xsd:simpleType> <xsd:restriction base="xsd:int"/> </xsd:simpleType> </xsd:list> </xsd:simpleType></pre>	<p>Nível de mapeamento 1.0:</p> <p>Não-suportado</p> <p>Nível de mapeamento 1.1 e superior:</p> <p>PIC X(255)</p>
<pre><xsd:simpleType> <xsd:union memberTypes="xsd:int xsd:string"/> </xsd:simpleType></pre>	<p>Nível de mapeamento 1.0:</p> <p>Não-suportado</p> <p>Nível de mapeamento 1.1 e 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>em que o comprimento não está definido.</p>	<p>Nível de mapeamento 1.0:</p> <p>Não-suportado</p> <p>Nível de Mapeamento 1.1:</p> <p>PIC X(</p> <p>y</p> <p>)</p> <p>em que $y = 4 \times (\text{ceil}(z / 3))$. $\text{ceil}(x)$ é o menor número inteiro maior ou igual a x.</p> <p>Nível de mapeamento 1.2 e posterior:</p> <p>PIC X(</p> <p>z</p> <p>)</p> <p>onde o comprimento é fixo.</p> <p>PIC X(16)</p> <p>Onde o comprimento não está definido. O campo contém o nome de 16 bytes do contêiner que armazena os dados binários.</p>

Tipo Simples de Esquema	Descrição de Dados COBOL
<pre><xsd:simpleType> <xsd:restriction base="xsd:float"> </xsd:restriction> </xsd:simpleType></pre>	<p>Nível de mapeamento 1.1 e abaixo:</p> <p>PIC X(32)</p> <p>Nível de mapeamento 1.2 e posterior:</p> <p>COMP-1</p> <p>Nota: A representação de dados IBM Hexadecimal Floating Point (HFP) não é exatamente a mesma que a representação IEEE-754-1985 usada para XML. Alguns valores podem não ser convertidos exatamente de uma representação para a outra. Alguns valores extremamente grandes ou pequenos podem não ser válidos para tipos de dados <code>xsd:float</code>. Alguns valores podem perder a precisão quando convertidos para ou a partir da representação HFP. Se conversões precisas forem importantes, considere substituir o uso de tipos de dados COMP-1 por alternativas de precisão fixas.</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:double"> </xsd:restriction> </xsd:simpleType></pre>	<p>Nível de mapeamento 1.1 e abaixo:</p> <p>PIC X(32)</p> <p>Nível de mapeamento 1.2 e posterior:</p> <p>COMP-2</p> <p>Nota: A representação de dados IBM Hexadecimal Floating Point (HFP) não é exatamente a mesma que a representação IEEE-754-1985 usada para XML. Alguns valores podem não ser convertidos exatamente de uma representação para a outra. Alguns valores extremamente grandes ou pequenos podem não ser válidos para os tipos de dados <code>xsd:double</code>. Alguns valores podem perder a precisão quando convertidos para ou a partir da representação HFP. Se conversões precisas forem importantes, considere substituir o uso de tipos de dados COMP-2 por alternativas de precisão fixas.</p>

Alguns dos tipos de esquema mostrados no mapa da tabela para um formato COBOL de COMP-5 SYNC ou de DISPLAY, dependendo dos valores (se houver) que são especificados nas máscaras `minInclusive` e `maxInclusive`:

- Para tipos sinalizados (`short`, `int` e `long`), DISPLAY é usado quando o seguinte é especificado:

```
<xsd:minInclusive value="-
a"/>
<xsd:maxInclusive value="
a
"/>
```

em que *a* é uma sequência de '9'.

- Para tipos não sinalizados (`unsignedShort`, `unsignedInt` e `unsignedLong`), DISPLAY é usado quando o seguinte é especificado:

```

<xsd:minInclusive value="0"/>
<xsd:maxInclusive value="
a
"/>

```

em que a é uma sequência de '9'.

Quando qualquer outro valor é especificado, ou nenhum valor é especificado, COMP-5 SYNC é usado.

Mapeamento de C e C++ para o Esquema XML:

Os programas utilitários DFHLS2SC e DFHLS2WS suportam mapeamentos entre tipos de dados C e C++ e definições de esquema XML.

Os nomes C e C++ são convertidos em nomes XML de acordo com as regras a seguir:

1. Os caracteres que não são válidos nos nomes de elementos XML são substituídos por 'X'.
Por exemplo, `monthly-total` se torna `monthlyXtotal`.
2. Nomes duplicados se tornam exclusivos pela adição de um ou mais dígitos numéricos.
Por exemplo, duas instâncias de `year` se tornam `year` e `year1`.

DFHLS2SC e DFHLS2WS mapeiam tipos de dados C e C++ para elementos de esquema de acordo com a tabela a seguir. Os tipos C e C++ que não são mostrados na tabela não são suportados pelo DFHLS2SC ou DFHLS2WS. O qualificador `_Packed` é suportado para estruturas. Estas restrições se aplicam:

- Os arquivos de cabeçalho devem conter uma instância `struct` de nível superior.
- Não é possível declarar um tipo de estrutura que contenha ele mesmo como um membro.
- Os seguintes tipos de dados C e C++ não são suportados:
`decimal`
`long double`
`wchar_t` (somente C++)
- Os caracteres a seguir são ignorados se estão presentes no arquivo de cabeçalho.

Especificadores de classe de armazenamento:

```

auto
register
static
extern
mutable

```

Qualificadores

```

const
volatile
_Export (somente C++)

```

Especificações de funções

```

inline (somente C++)
virtual (somente C++)

```

Valores iniciais

- O arquivo de cabeçalho não deve conter estes itens:
`Unions`
`Declarações de classe`
`Tipos de dados de enumeração`

Variáveis de tipo de ponteiro

Declarações de modelo

Macros predefinidas; isto é, macros com nomes que iniciam e terminam com dois caracteres de sublinhado (`__`)

A sequência de continuação de linha (um símbolo `\` que é imediatamente seguido por um caractere de nova linha)

Declaradores de função de protótipo

Diretivas de pré-processadores

Campos de bits

A palavra-chave `__cdecl` (ou `_cdecl`) (somente C++)

- O programador do aplicativo deve utilizar um compilador de 32 bits para assegurar que um `int` seja mapeado para 4 bytes.
- As palavras-chave reservadas por C++ a seguir não são suportadas:
 - `explicit`
 - `using`
 - `namespace`
 - `typename`
 - `typeid`
- Se o parâmetro **MAPPING-LEVEL** for configurado como 1.2 ou superior e o parâmetro **CHAR-VARYING** for configurado como `NULL`, as matrizes de caracteres serão mapeadas para um `xsd:string` e serão processadas como sequências com terminação nula.
- Se o parâmetro **MAPPING-LEVEL** for configurado como 1.2 ou superior e o parâmetro **CHAR-VARYING** for configurado como `BINARY`, as matrizes de caracteres serão mapeadas para `xsd:base64Binary` e serão processadas como dados binários.
- Se o parâmetro **MAPPING-LEVEL** for configurado como 1.2 ou superior e o parâmetro **CHAR-VARYING** for configurado como `COLLAPSE`, `<xsd:whiteSpace value="collapse"/>` será gerado para sequências.

Tipo de Dados C e C++	simpleType de Esquema
<code>char[z]</code>	<pre><xsd:simpleType> <xsd:restriction base="xsd:string"> <xsd:length value=" z"/> </xsd:restriction> </xsd:simpleType></pre>
<code>char16_t[n]</code>	<p>No nível de mapeamento 4.0 e superior:</p> <pre><xsd:simpleType> <xsd:restriction base="xsd:string"> <xsd:maxLength value="n"/> <xsd:whiteSpace value="preserve"/> </xsd:restriction> </xsd:simpleType></pre> <p>No tempo de execução, CICS preenche o campo de estrutura de dados do aplicativo com dados UTF-16.</p>

Tipo de Dados C e C++	simpleType de Esquema
char[8] Suportado no nível de mapeamento 3.0 e superior quando DATETIME=PACKED15	<pre><xsd:simpleType> <xsd:restriction base="xsd:dateTime" </xsd:restriction> </xsd:simpleType></pre> <p>O formato do registro de data e hora é CICS ABSTIME.</p>
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 (somente C++)	<pre><xsd:simpleType> <xsd:restriction base="xsd:boolean"> </xsd:restriction> </xsd:simpleType></pre>
float Suportado no nível de mapeamento 1.2 e superior	<pre><xsd:simpleType> <xsd:restriction base="xsd:float"> </xsd:restriction> </xsd:simpleType></pre> <p>Nota: A representação de dados do IBM Hexadecimal Floating Point (HFP) não é exatamente a mesma que a representação IEEE-754-1985 usada para XML. Alguns valores podem não ser convertidos exatamente de uma representação para a outra. Alguns valores extremamente grandes ou pequenos podem não ser válidos para os tipos de dados xsd:float. Alguns valores podem perder precisão quando convertidos para ou a partir de uma representação HFP. Se conversões precisas forem importantes, considere substituir o uso de tipos de dados flutuantes por alternativas de precisão fixas.</p>

Tipo de Dados C e C++	simpleType de Esquema
double Suportado no nível de mapeamento 1.2 e superior	<pre><xsd:simpleType> <xsd:restriction base="xsd:double"> </xsd:restriction> </xsd:simpleType></pre> <p>Nota: A representação de dados do IBM Hexadecimal Floating Point (HFP) não é exatamente a mesma que a representação IEEE-754-1985 usada para XML. Alguns valores podem não ser convertidos exatamente de uma representação para a outra. Alguns valores extremamente grandes ou pequenos podem não ser válidos para os tipos de dados xsd:double. Alguns valores podem perder precisão quando convertidos para ou a partir de uma representação HFP. Se conversões precisas forem importantes, considere substituir o uso de tipos duplos de dados com alternativas de precisão fixas.</p>

Esquema XML para Mapeamento C e C++:

Os programas utilitários DFHSC2LS e DFHWS2LS suportam mapeamentos entre as definições de esquema XML que são incluídas em cada descrição de serviços da web e tipos de dados C e C++.

Os assistentes do CICS geram nomes exclusivos e válidos para variáveis C e C++ a partir de nomes de elemento de esquema usando as regras a seguir:

1. Caracteres diferentes de A-Z, a-z, 0-9 ou _ são substituídos por ' X' inicial.
Por exemplo, monthly-total se torna monthlyXtotal.
2. Se o primeiro caractere não for um caractere alfabético, ele será substituído por um ' X' inicial.
Por exemplo, _monthlysummary se torna Xmonthlysummary.
3. Se o nome de elemento do esquema for maior do que 50 caracteres, ele será truncado para esse comprimento.
4. Nomes duplicados no mesmo escopo se tornam exclusivos pela adição de um ou mais dígitos numéricos.
Por exemplo, duas instâncias de year se tornam year e year1.
5. Cinco caracteres são reservados para as sequências _cont ou _num, que são usadas quando o esquema especifica que a variável possui cardinalidade variada; ou seja, quando minOccurs e maxOccurs são especificados em um xsd:element.
Para obter mais informações, consulte “Matrizes de Elementos Variáveis” na página 347.
6. Para atributos, as regras anteriores são aplicadas ao nome de elemento. O prefixo attr_ é incluído no nome de elemento e é seguido por _value ou _exist. Se o comprimento total for maior do que 28 caracteres, o nome de elemento será truncado.
O atributo anulável possui regras especiais. O prefixo attr_ é incluído, mas nil_ também é incluído no início do nome de elemento. O nome de elemento é seguido por _value. Se o comprimento total for maior do que 28 caracteres, o nome de elemento será truncado.

O comprimento total do nome resultante é 57 caracteres ou menos.

DFHSC2LS e DFHWS2LS mapeiam tipos de esquema para tipos de dados C e C++ de acordo com a tabela a seguir. As regras a seguir também se aplicam:

- Se o parâmetro **MAPPING-LEVEL** for configurado como 1.2 ou superior e o parâmetro **CHAR-VARYING** for configurado como NULL, os dados de caractere de comprimento variável serão mapeados para sequências com terminação nula e um caractere extra será alocado para o terminador NULL.
- Se o parâmetro **MAPPING-LEVEL** for configurado como 1.2 ou superior e o parâmetro **CHAR-VARYING** for configurado como YES, os dados de caractere de comprimento variável serão mapeados para dois elementos relacionados: um campo de comprimento e um campo de dados.

simpleType de Esquema	Tipo de Dados C e C++
<pre><xsd:simpleType> <xsd:restriction base="xsd:anyType"> </xsd:restriction> </xsd:simpleType></pre>	<p>Nível de mapeamento 2.0 e abaixo:</p> <p>Não-suportado</p> <p>Nível de mapeamento 2.1 e superior:</p> <p>Suportado</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:anySimpletype"> </xsd:restriction> </xsd:simpleType></pre>	<p>Nível de mapeamento 1.0:</p> <p>Não-suportado</p> <p>Nível de mapeamento 1.1 e superior:</p> <p>char[255]</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd: type"> <xsd:length value=" z"/> </xsd:restriction> </xsd:simpleType></pre> <p>em que <i>type</i> é um de:</p> <ul style="list-style-type: none"> string normalizedString token Name NMTOKEN language NCName ID IDREF ENTITY hexBinary 	<p>Todos os níveis de mapeamento:</p> <p>char[</p> <p>z</p> <p>]</p>

simpleType de Esquema	Tipo de Dados C e C++
<pre><xsd:simpleType> <xsd:restriction base="xsd: type"> <xsd:length value=" z"/> </xsd:restriction> </xsd:simpleType></pre> <p>em que <i>type</i> é um de:</p> <ul style="list-style-type: none"> string normalizedString token Name NMTOKEN language NCName ID IDREF ENTITY 	<p>Quando CCSID=1200 no nível de mapeamento 4.0 e superior:</p> <pre>char16_t[z]</pre>
<pre><xsd:simpleType> <xsd:restriction base="xsd: type"> </xsd:restriction> </xsd:simpleType></pre> <p>em que <i>type</i> é um de:</p> <ul style="list-style-type: none"> duration date decimal time gDay gMonth gYear gMonthDay gYearMonth 	<p>Todos os níveis de mapeamento:</p> <pre>char[32]</pre>
<pre><xsd:simpleType> <xsd:restriction base="xsd:dateTime"> </xsd:restriction> </xsd:simpleType></pre>	<p>Nível de mapeamento 1.2 e abaixo:</p> <pre>char[32]</pre> <p>Nível de mapeamento 2.0 e superior:</p> <pre>char[40]</pre> <p>Nível de mapeamento 3.0 e superior:</p> <pre>char[8]</pre> <p>O formato do registro de data e hora é CICS ABSTIME.</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:byte"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos os níveis de mapeamento:</p> <pre>signed char</pre>

simpleType de Esquema	Tipo de Dados C e C++
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedByte"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos os níveis de mapeamento:</p> <p>char</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:short"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos os níveis de mapeamento:</p> <p>short</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedShort"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos os níveis de mapeamento:</p> <p>unsigned short</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:integer"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos os níveis de mapeamento:</p> <p>char[33]</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:int"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos os níveis de mapeamento:</p> <p>int</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedInt"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos os níveis de mapeamento:</p> <p>unsigned int</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:long"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos os níveis de mapeamento:</p> <p>long long</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedLong"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos os níveis de mapeamento:</p> <p>unsigned long long</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:boolean"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos os níveis de mapeamento:</p> <p>bool (somente C++) short (somente C)</p>

simpleType de Esquema	Tipo de Dados C e C++
<pre><xsd:simpleType> <xsd:list> <xsd:simpleType> <xsd:restriction base="xsd:int"/> </xsd:simpleType> </xsd:list> </xsd:simpleType></pre>	<p>Nível de mapeamento 1.0:</p> <p>Não-suportado</p> <p>Nível de mapeamento 1.1 e superior:</p> <p>char[255]</p>
<pre><xsd:simpleType> <xsd:union memberTypes="xsd:int xsd:string"/> </xsd:simpleType></pre>	<p>Nível de mapeamento 1.0:</p> <p>Não-suportado</p> <p>Nível de mapeamento 1.1 e 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>onde o comprimento não está definido</p>	<p>Nível de mapeamento 1.1 e abaixo:</p> <p>char[y]</p> <p>where $y = 4 \times (\text{ceil}(z / 3))$. $\text{ceil}(x)$ is the smallest integer greater than or equal to x.</p> <p>Nível de mapeamento 1.2 e posterior:</p> <p>char[z]</p> <p>onde o comprimento é fixo.</p> <p>char[16]</p> <p>é o nome do contêiner que armazena os dados binários quando o comprimento não está definido.</p>

simpleType de Esquema	Tipo de Dados C e C++
<pre><xsd:simpleType> <xsd:restriction base="xsd:float"> </xsd:restriction> </xsd:simpleType></pre>	<p>Nível de mapeamento 1.1 e abaixo:</p> <p style="text-align: center;">char[32]</p> <p>Nível de mapeamento 1.2 e posterior:</p> <p style="text-align: center;">float(*)</p> <p>Nota: A representação de dados do IBM Hexadecimal Floating Point (HFP) não é exatamente a mesma que a representação IEEE-754-1985 usada para XML. Alguns valores podem não ser convertidos exatamente de uma representação para a outra. Alguns valores extremamente grandes ou pequenos podem não ser válidos para os tipos de dados xsd:float. Alguns valores podem perder precisão quando convertidos para ou a partir de uma representação HFP. Se conversões precisas forem importantes, considere substituir o uso de tipos de dados flutuantes por alternativas de precisão fixas.</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:double"> </xsd:restriction> </xsd:simpleType></pre>	<p>Nível de mapeamento 1.0 e abaixo:</p> <p style="text-align: center;">char[32]</p> <p>Nível de mapeamento 1.2 e posterior:</p> <p style="text-align: center;">double(*)</p> <p>Nota: A representação de dados do IBM Hexadecimal Floating Point (HFP) não é exatamente a mesma que a representação IEEE-754-1985 usada para XML. Alguns valores podem não ser convertidos exatamente de uma representação para a outra. Alguns valores extremamente grandes ou pequenos podem não ser válidos para os tipos de dados xsd:double. Alguns valores podem perder precisão quando convertidos para ou a partir de uma representação HFP. Se conversões precisas forem importantes, considere substituir o uso de tipos duplos de dados com alternativas de precisão fixas.</p>

Mapeamento de PL/I para Esquema XML:

Os programas utilitários DFHLS2SC e DFHLS2WS suportam os mapeamentos entre as estruturas de dados PL/I e as definições de esquema XML. Como o compilador PL/I Corporativo e compiladores PL/I mais antigos diferem, duas opções de linguagem são suportadas: PLI-ENTERPRISE e PLI-OTHER.

Os nomes PL/I são convertidos em nomes XML de acordo com as regras a seguir:

1. Os caracteres que não são válidos nos nomes de elementos XML são substituídos por ' x '.
Por exemplo, monthly\$total se torna monthlyxtotal.
2. Nomes duplicados se tornam exclusivos pela adição de um ou mais dígitos numéricos.
Por exemplo, duas instâncias de year se tornam year e year1.

DFHLS2SC e DFHLS2WS mapeiam os tipos de dados PL/I para os elementos de esquema, de acordo com a tabela a seguir. Os tipos PL/I que não são mostrados na tabela não são suportados por DFHLS2SC ou DFHLS2WS. As seguintes restrições também se aplicam:

- Os itens de dados com o atributo COMPLEX não são suportados.
- Os itens de dados com o atributo FLOAT são suportados em um nível de mapeamento 1.2 ou superior. Enterprise PL/I FLOAT IEEE não é suportado.
- As sequências DBCS puras VARYING e VARYINGZ são suportadas em um nível de mapeamento 1.2 ou superior.
- Os itens de dados que são especificados como DECIMAL(*p* , *q*) são suportados somente quando $p \geq q$
- Os itens de dados que são especificados como BINARY(*p* , *q*) são suportados somente quando $q = 0$.
- Se o atributo PRECISION for especificado para um item de dados, ele será ignorado.
- As sequências PICTURE não são suportadas.
- Itens de dados ORDINAL são tratados como tipos de dados FIXED BINARY(7) .
- Se o parâmetro **MAPPING-LEVEL** estiver configurado como 1.2 ou superior e o parâmetro **CHAR-VARYING** estiver configurado como NULL, as matrizes de caracteres serão mapeadas para um xsd:string e serão processadas como sequências com terminação nula; este mapeamento não se aplica para Enterprise PL/I.
- Se o parâmetro **MAPPING-LEVEL** for configurado como 1.2 ou superior e o parâmetro **CHAR-VARYING** for configurado como BINARY, as matrizes de caracteres serão mapeadas para xsd:base64Binary e serão processadas como dados binários.
- Se o parâmetro **MAPPING-LEVEL** for configurado como 1.2 ou superior e o parâmetro **CHAR-VARYING** for configurado como COLLAPSE, <xsd:whiteSpace value="collapse"/> será gerado para sequências.

DFHLS2SC e DFHLS2WS não implementam completamente os algoritmos de preenchimento de PL/I; portanto, você deve declarar os bytes de preenchimento explicitamente em sua estrutura de dados. DFHLS2SC e DFHLS2WS emitirão uma mensagem se eles detectarem que os bytes de preenchimento estão ausentes. Cada estrutura de nível superior deve iniciar em um limite de palavra dupla e cada byte na estrutura deve ser mapeado para o limite correto. Considere este fragmento de código:

```
3 FIELD1 FIXED BINARY(7),
3 FIELD2 FIXED BINARY(31),
3 FIELD3 FIXED BINARY(63);
```

Neste exemplo:

- FIELD1 possui 1 byte de comprimento e pode ser alinhado em qualquer limite.
- FIELD2 tem 4 bytes de comprimento e deve ser alinhado em um limite de palavra inteira.
- FIELD3 tem 8 bytes de comprimento e deve ser alinhado em um limite de palavra dupla.

O compilador Enterprise PL/I alinha os campos na ordem a seguir:

1. FIELD3 é alinhado primeiro porque ele tem os requisitos de limite mais fortes.
2. FIELD2 é alinhado no limite de palavra inteira imediatamente antes de FIELD3.
3. FIELD1 é alinhado no limite de byte imediatamente antes de FIELD3.

Finalmente, para que a estrutura inteira esteja alinhada em um limite de palavra inteira, o compilador insere três bytes de preenchimento imediatamente antes de FIELD1.

Como DFHLS2WS não insere bytes de preenchimento equivalentes, você deve declará-los explicitamente antes da estrutura ser processada por DFHLS2WS. Por exemplo:

```
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);
```

Como alternativa, é possível mudar a estrutura para declarar todos os campos como desalinhados e recompilar o aplicativo que usa a estrutura. Para obter mais informações sobre requisitos de alinhamento de memória estrutural PL/I, consulte *Referência de Linguagem Enterprise PL/I*.

Descrição de Dados PL/I	Schema
FIXED BINARY (n) em que $n \leq 7$	<code><xsd:simpleType> <xsd:restriction base="xsd:byte"/> </xsd:simpleType></code>
FIXED BINARY (n) em que $8 \leq n \leq 15$	<code><xsd:simpleType> <xsd:restriction base="xsd:short"/> </xsd:simpleType></code>
FIXED BINARY (n) em que $16 \leq n \leq 31$	<code><xsd:simpleType> <xsd:restriction base="xsd:int"/> </xsd:simpleType></code>
FIXED BINARY (n) em que $32 \leq n \leq 63$ Restrição: Somente PL/I c Corporativo	<code><xsd:simpleType> <xsd:restriction base="xsd:long"/> </xsd:simpleType></code>
UNSIGNED FIXED BINARY(n) em que $n \leq 8$ Restrição: Somente PL/I c Corporativo	<code><xsd:simpleType> <xsd:restriction base="xsd:unsignedByte"/> </xsd:simpleType></code>
UNSIGNED FIXED BINARY(n) em que $9 \leq n \leq 16$ Restrição: Somente PL/I c Corporativo	<code><xsd:simpleType> <xsd:restriction base="xsd:unsignedShort"/> </xsd:simpleType></code>
UNSIGNED FIXED BINARY(n) em que $17 \leq n \leq 32$ Restrição: Somente PL/I c Corporativo	<code><xsd:simpleType> <xsd:restriction base="xsd:unsignedInt"/> </xsd:simpleType></code>
UNSIGNED FIXED BINARY(n) em que $33 \leq n \leq 64$ Restrição: Somente PL/I c Corporativo	<code><xsd:simpleType> <xsd:restriction base="xsd:unsignedLong"/> </xsd:simpleType></code>
FIXED DECIMAL(n , m)	<code><xsd:simpleType> <xsd:restriction base="xsd:decimal"> <xsd:totalDigits value="n"/> <xsd:fractionDigits value="m"/> </xsd:restriction> </xsd:simpleType></code>

Descrição de Dados PL/I	Schema
<p>FIXED DECIMAL(15)</p> <p>Suportado no nível de mapeamento 3.0 e superior quando DATETIME=PACKED15</p>	<pre><xsd:simpleType> <xsd:restriction base="xsd:dateTime"> </xsd:restriction> </xsd:simpleType></pre> <p>O formato do registro de data e hora é CICS ABSTIME.</p>
<p>BIT(<i>n</i>)</p> <p>em que <i>n</i> é um múltiplo de 8. Outros valores não são suportados.</p>	<pre><xsd:simpleType> <xsd:restriction base="xsd:hexBinary"> <xsd:length value=" <i>m</i>" /> </xsd:restriction> </xsd:simpleType></pre> <p>em que $m = n / 8$</p>
<p>CHARACTER(<i>n</i>)</p> <p>VARYING e VARYINGZ também são suportados no nível de mapeamento 1.2 e superior.</p> <p>Restrição: VARYINGZ é suportado somente pelo PL/I Corporativo</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 e VARYINGZ também são suportados no nível de mapeamento 1.2 e superior.</p> <p>Restrição: VARYINGZ é suportado somente pelo Enterprise PL/I</p>	<p>Em um nível de mapeamento 1.0 e 1.1, em que $m = 2 * n$:</p> <pre><xsd:simpleType> <xsd:restriction base="xsd:hexBinary"> <xsd:length value=" <i>m</i>" /> </xsd:restriction> </xsd:simpleType></pre> <p>Em um nível de mapeamento de 1.2 ou 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>

Descrição de Dados PL/I	Schema
<p>WIDECHAR(n)</p> <p>Restrição: Somente PL/I c Corporativo</p>	<p>Em um nível de mapeamento 1.0 e 1.1, em que $m = 2 * n$:</p> <pre><xsd:simpleType> <xsd:restriction base="xsd:hexBinary"> <xsd:length value=" m"/> </xsd:restriction> </xsd:simpleType></pre> <p>Em um nível de mapeamento de 1.2 ou superior:</p> <pre><xsd:simpleType> <xsd:restriction base="xsd:hexBinary"> <xsd:length value=" n"/> </xsd:restriction> </xsd:simpleType></pre> <p>No nível de mapeamento 4.0 e superior, o CICS preenche o campo de estrutura de dados do aplicativo com dados 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>Restrição: Apenas PL/I corporativo</p>	<pre><xsd:simpleType> <xsd:restriction base="xsd:byte"/> </xsd:simpleType></pre>
<p>BINARY FLOAT(n)</p> <p>where $n \leq 21$</p> <p>Suportado no nível de mapeamento 1.2 e superior.</p>	<pre><xsd:simpleType> <xsd:restriction base="xsd:float"> </xsd:restriction> </xsd:simpleType></pre>
<p>BINARY FLOAT(n)</p> <p>em que $21 < n \leq 53$</p> <p>Valores maiores que 53 não são suportados.</p> <p>Suportado no nível de mapeamento 1.2 e superior.</p>	<pre><xsd:simpleType> <xsd:restriction base="xsd:double"> </xsd:restriction> </xsd:simpleType></pre>
<p>DECIMAL FLOAT(n)</p> <p>em que $n \leq 6$</p> <p>Suportado no nível de mapeamento 1.2 e superior.</p> <p>Nota: A representação de dados do IBM Hexadecimal Floating Point (HFP) não é exatamente a mesma da representação IEEE-754-1985 usada para XML. Alguns valores podem não ser convertidos exatamente de uma representação para a outra. Alguns valores extremamente grandes ou pequenos podem não ser válidos para os tipos de dados xsd:float. Alguns valores podem perder precisão quando convertidos para ou a partir de uma representação HFP. Se conversões precisas forem importantes, considere substituir o uso de tipos de dados DECIMAL FLOAT por alternativas de precisão fixas.</p>	<pre><xsd:simpleType> <xsd:restriction base="xsd:float"> </xsd:restriction> </xsd:simpleType></pre>

Descrição de Dados PL/I	Schema
<p>DECIMAL FLOAT(<i>n</i>) em que $6 < n \leq 16$</p> <p>Valores maiores que 16 não são suportados.</p> <p>Suportado no nível de mapeamento 1.2 e superior.</p> <p>Nota: A representação de dados do IBM Hexadecimal Floating Point (HFP) não é exatamente a mesma da representação IEEE-754-1985 usada para XML. Alguns valores podem não ser convertidos exatamente de uma representação para a outra. Alguns valores extremamente grandes ou pequenos podem não ser válidos para os tipos de dados <code>xsd:double</code>. Alguns valores podem perder precisão quando convertidos para ou a partir de uma representação HFP. Se conversões precisas forem importantes, considere substituir o uso de tipos de dados DECIMAL FLOAT por alternativas de precisão fixas.</p>	<pre><xsd:simpleType> <xsd:restriction base="xsd:double"> </xsd:restriction> </xsd:simpleType></pre>

Esquema XML para Mapeamento de PL/I:

Os programas utilitários DFHSC2LS e DFHWS2LS suportam mapeamentos entre definições de esquema XML e estruturas de dados PL/I. Como o compilador PL/I Corporativo e compiladores PL/I mais antigos diferem, duas opções de linguagem são suportadas: PLI-ENTERPRISE e PLI-OTHER.

Regras para mapear nomes de elemento de esquema para PL/I

Os assistentes do CICS geram nomes exclusivos e válidos para variáveis PL/I a partir dos nomes de elemento de esquema usando as regras a seguir:

1. Caracteres diferentes de A-Z, a-z, 0-9, @, #, _ ou \$ são substituídos por ' X' inicial.

Por exemplo, `monthly-total` se torna `monthlyXtotal`.

É possível usar o parâmetro **MAPPING-OVERRIDES** para mudar o modo como outros caracteres são manipulados. Por exemplo, se você configurar o valor **HYPHENS-AS-UNDERSCORES**, qualquer hífen no XML será convertido em um sublinhado em vez de um X. Por exemplo, `monthly-total` se torna `monthly_total`.

2. Se o esquema especificar que a variável possui variação de cardinalidade (ou seja, os atributos `minOccurs` e `maxOccurs` são especificados com valores diferentes no `xsd:element`) e o nome de elemento do esquema for maior que 24 caracteres, ele será truncado para esse comprimento.

Se o esquema especificar que a variável possui cardinalidade fixa e o nome de elemento do esquema for maior do que 29 caracteres, ele será truncado para esse comprimento.

3. Nomes duplicados no mesmo escopo se tornam exclusivos pela adição de um ou mais dígitos numéricos na segunda instância e nas instâncias subsequentes do nome.

Por exemplo, três instâncias de `year` se tornam `year`, `year1` e `year2`.

4. Cinco caracteres são reservados para as sequências `_cont` ou `_num`, que são usadas quando o esquema especifica que a variável possui variação de cardinalidade; ou seja, quando os atributos `minOccurs` e `maxOccurs` são especificados com valores diferentes.

Para obter mais informações, consulte “Matrizes de Elementos Variáveis” na página 347.

5. Para atributos, as regras anteriores são aplicadas ao nome de elemento. O prefixo attr- é incluído no nome de elemento e é seguido por -value ou -exist. Se o comprimento total for maior do que 28 caracteres, o nome de elemento será truncado. Para obter mais informações, consulte “Suporte para Atributos XML” na página 352.

O atributo anulável possui regras especiais. O prefixo attr- é incluído, mas nil- também é incluído no início do nome de elemento. O nome de elemento é seguido por -value. Se o comprimento total for maior do que 28 caracteres, o nome de elemento será truncado.

O comprimento total do nome resultante é 31 caracteres ou menos.

Regras para mapear tipos de esquema para PL/I

DFHSC2LS e DFHWS2LS mapeiam tipos de esquema para tipos de dados PL/I de acordo com a tabela a seguir. Além disso, observe os seguintes pontos:

- Se o parâmetro **MAPPING-LEVEL** for configurado como 1.2 ou superior e o parâmetro **CHAR-VARYING** for configurado como NULL, os dados de caractere de comprimento variável serão mapeados para sequências com terminação nula e um caractere extra será alocado para o terminador NULL.
- Se o parâmetro **MAPPING-LEVEL** for configurado como 1.2 ou superior e o parâmetro **CHAR-VARYING** não for especificado, por padrão, os dados de caractere de comprimento variável serão mapeados para um tipo de dados VARYINGZ para Enterprise PL/I e para o tipo de dados VARYING para Outro PL/I.
- Dados binários de comprimento variável são mapeados para um tipo de dados VARYING se são menores que 32.768 bytes e para um contêiner se são maiores que 32.768 bytes.

Schema	Descrição de Dados PL/I
<pre><xsd:simpleType> <xsd:restriction base="xsd:anyType"> </xsd:restriction> </xsd:simpleType></pre>	<p>Nível de mapeamento 2.0 e abaixo:</p> <p>Não-suportado</p> <p>Nível de mapeamento 2.1 e superior:</p> <p>Suportado</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:anySimpleType"> </xsd:restriction> </xsd:simpleType></pre>	<p>Nível de mapeamento 1.1 e superior: CHAR(255)</p>

Schema	Descrição de Dados PL/I
<pre><xsd:simpleType> <xsd:restriction base="xsd: type"> <xsd:maxLength value=" z"/> <xsd:whiteSpace value="preserve"/> </xsd:restriction> </xsd:simpleType></pre> <p>em que <i>type</i> é um de:</p> <ul style="list-style-type: none"> string normalizedString token Name NMTOKEN language NCName ID IDREF ENTITY 	<p>Todos os níveis de mapeamento: CHARACTER(z)</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd: type"> <xsd:length value=" z"/> </xsd:restriction> </xsd:simpleType></pre> <p>em que <i>type</i> é um de:</p> <ul style="list-style-type: none"> string normalizedString token Name NMTOKEN language NCName ID IDREF ENTITY 	<p>Quando CCSID=1200 no nível de mapeamento 4.0 e superior:</p> <p>WIDECHAR(</p> <p>z</p> <p>)</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd: type"> </xsd:restriction> </xsd:simpleType></pre> <p>em que <i>type</i> é um de:</p> <ul style="list-style-type: none"> duration date time gDay gMonth gYear gMonthDay gYearMonth 	<p>Todos os níveis de mapeamento: CHAR(32)</p>

Schema	Descrição de Dados PL/I
<pre><xsd:simpleType> <xsd:restriction base="xsd:dateTime"> </xsd:restriction> </xsd:simpleType></pre>	<p>Nível de mapeamento 1.2 e abaixo:</p> <p>CHAR(32)</p> <p>Nível de mapeamento 2.0 e superior:</p> <p>CHAR(40)</p> <p>Nível de mapeamento 3.0 e superior:</p> <p>FIXED DECIMAL(15)</p> <p>O formato do registro de data e hora é CICS ABSTIME.</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:hexBinary"> <xsd:length value=" y"/> </xsd:restriction> </xsd:simpleType></pre>	<p>Nível de mapeamento 1.1 e abaixo:</p> <p>BIT(z)</p> <p>em que $z = 8 \times y$ e $z < 4095$ bytes.</p> <p>CHAR(z)</p> <p>em que $z = 8 \times y$ e $z > 4095$ bytes.</p> <p>Níveis de mapeamento 1.2 e superior:</p> <p>CHAR(y)</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:byte"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos os níveis de mapeamento:</p> <p>Enterprise PL/I SIGNED FIXED BINARY (7)</p> <p>Outro PL/I FIXED BINARY (7)</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedByte"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos os níveis de mapeamento:</p> <p>Enterprise PL/I UNSIGNED FIXED BINARY (8)</p> <p>Outro PL/I FIXED BINARY (8)</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:short"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos os níveis de mapeamento:</p> <p>Enterprise PL/I SIGNED FIXED BINARY (15)</p> <p>Outro PL/I FIXED BINARY (15)</p>

Schema	Descrição de Dados PL/I
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedShort"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos os níveis de mapeamento:</p> <p>Enterprise PL/I UNSIGNED FIXED BINARY (16)</p> <p>Outro PL/I FIXED BINARY (16)</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:integer"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos os níveis de mapeamento:</p> <p>Enterprise PL/I FIXED DECIMAL(31,0)</p> <p>Outro PL/I FIXED DECIMAL (15,0)</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:int"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos os níveis de mapeamento:</p> <p>Enterprise PL/I SIGNED FIXED BINARY (31)</p> <p>Outro PL/I FIXED BINARY (31)</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedInt"> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos os níveis de mapeamento:</p> <p>Enterprise PL/I UNSIGNED FIXED BINARY(32)</p> <p>Outro PL/I BIT(64)</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:long"> </xsd:restriction> </xsd:simpleType></pre>	<p>Nível de mapeamento 1.1 e abaixo:</p> <p>Enterprise PL/I SIGNED FIXED BINARY(63)</p> <p>Nível de mapeamento 1.2 e posterior:</p> <p>Enterprise PL/I CHAR(y)</p> <p>em que y é um comprimento fixo menor do que 16 MB.</p> <p>Todos os níveis de mapeamento:</p> <p>Outro PL/I BIT(64)</p>

Schema	Descrição de Dados PL/I
<pre><xsd:simpleType> <xsd:restriction base="xsd:unsignedLong"> </xsd:restriction> </xsd:simpleType></pre>	<p>Nível de mapeamento 1.1 e abaixo:</p> <p>Enterprise PL/I UNSIGNED FIXED BINARY (64)</p> <p>Nível de mapeamento 1.2 e posterior:</p> <p>Enterprise PL/I CHAR(<i>y</i>) em que <i>y</i> é um comprimento fixo menor do que 16 MB.</p> <p>Todos os níveis de mapeamento:</p> <p>Outro PL/I BIT(64)</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:boolean"> </xsd:restriction> </xsd:simpleType></pre>	<p>Nível de mapeamento 1.1 e abaixo:</p> <p>Enterprise PL/I SIGNED FIXED BINARY (7)</p> <p>Outro PL/I FIXED BINARY (7)</p> <p>Nível de mapeamento 1.2 e posterior:</p> <p>Enterprise PL/I BIT(7) BIT(1)</p> <p>Outro PL/I BIT(7) BIT(1) em que BIT(7) é fornecido para alinhamento e BIT(1) contém o valor mapeado Booleano.</p>
<pre><xsd:simpleType> <xsd:restriction base="xsd:decimal"> <xsd:totalDigits value=" <i>n</i>" /> <xsd:fractionDigits value=" <i>m</i>" /> </xsd:restriction> </xsd:simpleType></pre>	<p>Todos os níveis de mapeamento: 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 os níveis de mapeamento: CHAR(255)</p>
<pre><xsd:simpleType> <xsd:union memberTypes="xsd:int xsd:string" /> </xsd:simpleType></pre>	<p>Todos os níveis de mapeamento: CHAR(255)</p>

Schema	Descrição de Dados 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>onde o comprimento não está definido</p>	<p>Nível de mapeamento 1.0:</p> <p>Não-suportado</p> <p>Nível de Mapeamento 1.1:</p> <p style="text-align: right;">CHAR(z)</p> <p>em que $z = 4 \times (\text{ceil}(y / 3))$. $\text{ceil}(x)$ é o menor número inteiro maior ou igual a x.</p> <p>Nível de mapeamento 1.2 e posterior:</p> <p style="text-align: right;">CHAR(y)</p> <p>onde o comprimento é fixo.</p> <p style="text-align: right;">CHAR(16)</p> <p>Onde o comprimento não está definido. O campo contém o nome de 16 bytes do contêiner que armazena os dados binários.</p>
<pre> <xsd:simpleType> <xsd:restriction base="xsd:float"> </xsd:restriction> </xsd:simpleType> </pre>	<p>Níveis de Mapeamento 1.0 e 1.1:</p> <p style="text-align: right;">CHAR(32)</p> <p>Nível de mapeamento 1.2 e posterior:</p> <p>Enterprise PL/I DECIMAL FLOAT(6) HEXADEC</p> <p>Outro PL/I DECIMAL FLOAT(6)</p> <p>Nota: A representação de dados do IBM Hexadecimal Floating Point (HFP) não é exatamente a mesma da representação IEEE-754-1985 usada para XML. Alguns valores podem não ser convertidos exatamente de uma representação para a outra. Alguns valores extremamente grandes ou pequenos podem não ser válidos para os tipos de dados <code>xsd:float</code>. Alguns valores podem perder precisão quando convertidos para ou a partir de uma representação HFP. Se conversões precisas forem importantes, considere substituir o uso de tipos de dados DECIMAL FLOAT por alternativas de precisão fixas.</p>

Schema	Descrição de Dados PL/I
<pre><xsd:simpleType> <xsd:restriction base="xsd:double"> </xsd:restriction> </xsd:simpleType></pre>	<p>Níveis de Mapeamento 1.0 e 1.1:</p> <p>CHAR(32)</p> <p>Nível de mapeamento 1.2 e posterior:</p> <p>Enterprise PL/I DECIMAL FLOAT(16) HEXADEC</p> <p>Outro PL/I DECIMAL FLOAT(16)</p> <p>Nota: A representação de dados do IBM Hexadecimal Floating Point (HFP) não é exatamente a mesma da representação IEEE-754-1985 usada para XML. Alguns valores podem não ser convertidos exatamente de uma representação para a outra. Alguns valores extremamente grandes ou pequenos podem não ser válidos para os tipos de dados xsd:double. Alguns valores podem perder precisão quando convertidos para ou a partir de uma representação HFP. Se conversões precisas forem importantes, considere substituir o uso de tipos de dados DECIMAL FLOAT por alternativas de precisão fixas.</p>

Matrizes de Elementos Variáveis:

O XML pode conter uma matriz com números variados de elementos. Em geral, documentos WSDL e esquemas XML que contêm números variados de elementos não mapeiam eficientemente para uma única estrutura de dados de linguagem de alto nível. O CICS usa mapeamentos baseados em contêiner ou mapeamentos sequenciais para manipular números variados de elementos em XML.

Uma matriz com um número variado de elementos é representada no esquema XML usando os atributos minOccurs e maxOccurs na declaração de elemento:

- O atributo minOccurs especifica o número mínimo de vezes que o elemento pode ocorrer. Ela pode ter um valor 0 ou qualquer número inteiro positivo.
- O atributo maxOccurs especifica o número máximo de vezes que o elemento pode ocorrer. Ele pode ter um valor de qualquer número inteiro positivo maior ou igual ao valor do atributo minOccurs. Ele também pode ter um valor igual a unbounded, que indica que nenhum limite superior se aplica ao número de vezes que o elemento pode ocorrer.
- O valor padrão para ambos os atributos é 1.

Este exemplo denota uma sequência de 8 bytes que é opcional; ou seja, ela pode não ocorrer nunca ou ocorrer uma vez no XML do aplicativo ou na mensagem SOAP:

```
<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>
```

O exemplo a seguir denota uma sequência de 8 bytes que deve ocorrer pelo menos uma 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>
```

Em geral, documentos WSDL que contêm números variados de elementos não mapeiam eficientemente para uma única estrutura de dados de linguagem de alto nível. Portanto, para manipular esses casos, o CICS usa uma série de estruturas de dados conectadas que são transmitidas ao programa de aplicativo em uma série de contêineres. Essas estruturas são usadas como entrada e saída do aplicativo:

- Quando o CICS transforma XML em dados do aplicativo, ele preenche essas estruturas com os dados do aplicativo e o aplicativo os lê.
- Quando o CICS transforma os dados do aplicativo em XML, ele lê os dados do aplicativo nas estruturas que foram preenchidas pelo aplicativo.

O formato dessas estruturas de dados é explicado melhor com uma série de exemplos. O XML pode ser de uma mensagem SOAP ou de um aplicativo. Estes exemplos usam uma matriz de campos de 8 bytes simples. No entanto, o modelo suporta matrizes de tipos de dados complexos e matrizes de tipos de dados que contêm outras matrizes.

Número Fixo de Elementos

O primeiro exemplo ilustra um elemento que ocorre exatamente três vezes:

```
<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>
```

Neste exemplo, como o número de vezes que o elemento ocorre é conhecido antecipadamente, ele pode ser representado como uma matriz de comprimento fixo em uma declaração COBOL simples (ou o equivalente em outras linguagens):

```
05 component PIC X(8) OCCURS 3 TIMES
```

Número variado de elementos no nível de mapeamento 2 e abaixo

Este exemplo ilustra um elemento obrigatório que pode ocorrer de uma a cinco vezes:

```
<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>
```

A estrutura de dados principal contém uma declaração de dois campos. Quando o CICS transforma o XML em dados binários, o primeiro campo `component-num` contém o número de vezes que o elemento aparece no XML e o segundo campo, `component-cont`, contém o nome de um contêiner:

```
05 component-num PIC S9(9) COMP-5
05 component-cont PIC X(16)
```

Uma segunda estrutura de dados contém a declaração do elemento em si:

```
01 DFHWS-component
02 component PIC X(8)
```

Você deve examinar o valor de `component-num` (que conterá um valor no intervalo de 1 a 5) para descobrir quantas vezes o elemento ocorre. O conteúdo do elemento está no contêiner nomeado em `component-cont`; o contêiner contém uma matriz de elementos, em que cada elemento é mapeado pela estrutura de dados `DFHWS-component`.

Se `minOccurs="0"` e `maxOccurs="1"`, o elemento é opcional. Para processar a estrutura de dados em seu programa de aplicativo, você deve examinar o valor de `component-num`:

- Se for zero, a mensagem não possui elemento do componente e o conteúdo de `component-cont` é indefinido.
- Se for um, o elemento do componente está no contêiner nomeado em `component-cont`.

O conteúdo do contêiner é mapeado pela estrutura de dados `DFHWS-component`.

Nota: Se a mensagem SOAP consistir em um único elemento recorrente, o `DFHWS2LS` gerará duas estruturas de linguagem. A estrutura de linguagem principal contém o número de elementos na matriz e o nome de um contêiner que contém a matriz de elementos. A segunda estrutura de linguagem mapeia uma única instância do elemento recorrente.

Número variado de elementos no nível de mapeamento 2.1 e superior

No nível de mapeamento 2.1 e acima, é possível usar o parâmetro **INLINE-MAXOCCURS-LIMIT** nos assistentes do CICS. O parâmetro **INLINE-MAXOCCURS-LIMIT** especifica a maneira como números variados de elementos são manipulados. As opções de mapeamento para números variados de elementos são mapeamento baseado em contêiner, descrito em “Número variado de elementos no nível de mapeamento 2 e abaixo” na página 348, ou mapeamento sequencial. O *value* deste parâmetro pode ser um número inteiro positivo no intervalo de 0 a 32767:

- O valor padrão de **INLINE-MAXOCCURS-LIMIT** é 1, o qual assegura que elementos opcionais sejam mapeados sequencialmente.
- Um valor 0 para o parâmetro **INLINE-MAXOCCURS-LIMIT** evita o mapeamento sequencial.
- Se `maxOccurs` for menor ou igual ao valor de **INLINE-MAXOCCURS-LIMIT**, o mapeamento sequencial será usado.
- Se `maxOccurs` for maior do que o valor de **INLINE-MAXOCCURS-LIMIT**, o mapeamento baseado em contêiner será usado.

O mapeamento de números variados de elementos sequenciais resulta na geração de uma matriz, como acontece com o exemplo de ocorrência fixa acima, e de um contador. O campo `component-num` indica quantas instâncias do elemento estão

presentes e elas são apontadas pela matriz. Para o exemplo mostrado em “Número variado de elementos no nível de mapeamento 2 e abaixo” na página 348, quando **INLINE-MAXOCCURS-LIMIT** é menor ou igual a 5, a estrutura de dados gerada conforme a seguir:

```
05 component-num PIC S9(9) COMP-5 SYNC.  
05 component OCCURS 5 PIC X(8).
```

O primeiro campo, component-num, é idêntico ao de saída para o mapeamento baseado em contêiner de exemplo na seção anterior. O segundo campo contém uma matriz de comprimento 5 que é grande o suficiente para conter o número máximo de elementos que podem ser gerados.

O mapeamento sequencial difere do mapeamento baseado em contêiner, que armazena o número de ocorrências do elemento e o nome do contêiner no qual os dados são colocados, porque ele armazena todos os dados no contêiner atual. O armazenamento dos dados no contêiner atual geralmente irá melhorar o desempenho e tornar o mapeamento sequencial preferível.

Matrizes de Variáveis Aninhadas

Os documentos WSDL e esquemas XML complexos podem conter elementos variavelmente recorrentes, que por sua vez contêm os elementos variavelmente recorrentes. Nesse caso, a estrutura descrita se estende além dos dois níveis descritos nos exemplos.

Este exemplo ilustra um elemento opcional chamado <component2> que está aninhado em um elemento obrigatório chamado <component1>, em que o elemento obrigatório pode ocorrer de uma a cinco vezes:

```
<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>
```

A estrutura de dados de nível superior é exatamente a mesma que nos exemplos anteriores:

```
05 component1-num PIC S9(9) COMP-5  
05 component1-cont PIC X(16)
```

No entanto, a segunda estrutura de dados contém estes elementos:

```
01 DFHWS-component1  
02 component2-num PIC S9(9) COMP-5  
02 component2-cont PIC X(16)
```

Uma estrutura de terceiro nível contém estes elementos:

```
01 DFHWS-component2  
02 component2 PIC X(8)
```

O número de ocorrências do elemento <component1> mais externo está em component1-num.

O contêiner nomeado em component1-cont contém uma matriz com esse número de instâncias da segunda estrutura de dados DFHWS-component1.

Cada instância de component2-cont nomeia um contêiner diferente, cada um dos quais contém a estrutura de dados mapeada pela estrutura de terceiro nível DFHWS-component2.

Para ilustrar essa estrutura, considere o fragmento de XML que corresponde ao exemplo:

```
<component1><component2>
string1
</component2></component1>
<component1><component2>
string2
</component2></component1>
<component1></component1>
```

<component1> ocorre três vezes. Cada um dos dois primeiros contém uma instância de <component2>; a terceira instância não.

Na estrutura de dados de nível superior, component1-num contém um valor igual a 3. O contêiner nomeado em component1-cont possui três instâncias de DFHWS-component1 :

1. Na primeira, component2-num possui um valor 1 e o contêiner nomeado em component2-cont contém *string1*.
2. Na segunda, component2-num possui um valor 1 e o contêiner nomeado em component2-cont contém *string2*.
3. Na terceira, component2-num possui um valor 0 e o conteúdo de component2-cont é indefinido.

Neste exemplo, a estrutura de dados completa é representada por quatro contêineres:

- A estrutura de dados raiz no contêiner DFHWS-DATA
- O contêiner nomeado em component1-cont
- Dois contêineres nomeados nas duas primeiras instâncias de component2-cont

Estruturas Opcionais e xsd:choice

DFHWS2LS e DFHSC2LS suportam o uso de maxOccurs e minOccurs nos elementos <xsd:sequence>, <xsd:choice> e <xsd:all> somente no nível de mapeamento 2.1 e acima, em que os atributos minOccurs e maxOccurs são configurados como minOccurs="0" e maxOccurs="1".

Os assistentes geram mapeamentos que tratam estes elementos como se cada elemento filho neles fosse opcional. Ao implementar um aplicativo com estes elementos, assegure que combinações inválidas de opções não sejam geradas pelo aplicativo. Cada um dos elementos possui seu próprio campo count na estrutura de linguagens gerada, todos estes campos devem ser configurados como "0" ou todos devem ser configurados como "1". Qualquer outra combinação de valores é inválida, exceto para elementos <xsd:choice>.

Elementos <xsd:choice> indicam que somente uma das opções no elemento pode ser usada. Isto é suportado em todos os níveis de mapeamento. Os assistentes

manipulam cada uma das opções em um <xsd:choice> como se estivesse em um elemento <xsd:sequence> com minOccurs="0" e maxOccurs="1". Tome cuidado ao implementar um aplicativo usando o elemento <xsd:choice> para assegurar que combinações inválidas de opções não sejam geradas pelo aplicativo. Cada um dos elementos possui seu próprio campo count na estrutura de linguagens gerada, exatamente um dos quais deve ser configurado como '1' e todos os outros devem ser configurados como '0'. Qualquer outra combinação de valores é inválida, exceto quando o elemento <xsd:choice> é em si opcional, nesse caso ele é válido para todos os campos a serem configurados como '0'.

Suporte para Atributos XML:

Os esquemas XML podem especificar atributos que são permitidos ou necessários no XML. Os utilitários do assistente CICS, DFHWS2LS e DFHSC2LS, ignoram atributos XML por padrão. Para processar atributos XML que são definidos no esquema XML, o valor do parâmetro **MAPPING-LEVEL** deve ser 1.1 ou superior.

Atributos Opcionais

Os atributos podem ser opcionais ou necessários e podem ser associados a qualquer elemento em uma mensagem SOAP ou XML para um aplicativo. Para cada atributo opcional definido no esquema, dois campos são gerados na estrutura de linguagem apropriada:

1. Uma sinalização de existência; este campo é tratado como um tipo de dados Booleano e geralmente tem 1 byte de comprimento.
2. Um valor; este campo é mapeado da mesma maneira que um elemento XML de tipo equivalente. Por exemplo, um atributo do tipo NMTOKEN é mapeado da mesma maneira que um elemento XML do tipo NMTOKEN.

Os campos de existência de atributo e de valor aparecem na estrutura de linguagem gerada antes do campo para o elemento ao qual eles estão associados. Atributos inesperados que aparecem no documento da instância são ignorados.

Por exemplo, considere a definição de atributo de esquema a seguir:

```
<xsd:attribute name="age" type="xsd:short"
use="optional" />
```

Este atributo opcional mapeia para a estrutura COBOL a seguir:

```
05 attr-age-exist PIC X DISPLAY
05 attr-age-value PIC S9999 COMP-5 SYNC
```

Processamento de Tempo de Execução de Atributos Opcionais

O processamento de tempo de execução a seguir ocorre para atributos opcionais:

- Se o atributo estiver presente, a sinalização de existência será configurada e o valor será mapeado.
- Se o atributo não estiver presente, o sinalizador de existência não será configurado.
- Se o atributo tiver um valor padrão e estiver presente, o valor será mapeado.
- Se o atributo tiver um valor padrão e não estiver presente, o valor padrão será mapeado.

Atributos opcionais que têm valores padrão são tratados como atributos necessários.

Quando o CICS transforma os dados em XML, o processamento de execução a seguir ocorre:

- Se a sinalização de existência estiver configurada, o atributo será transformado e incluído no XML.
- Se a sinalização de existência não estiver configurada, o atributo não será incluído no XML.

Atributos Necessários e o Processamento de Tempo de Execução

Para cada atributo que é necessário, somente o campo de valor é gerado na estrutura de linguagem apropriada.

Se o atributo estiver presente no XML, o valor será mapeado. Se o atributo não estiver presente, ocorrerá o seguinte processamento:

- Se o aplicativo for um provedor de serviço da web, o CICS gerará uma mensagem de falha SOAP indicando um erro na mensagem SOAP do cliente.
- Se o aplicativo for um solicitante de serviço da web, o CICS emitirá uma mensagem e retornará uma resposta de erro de conversão com um código RESP2 igual a 13 para o aplicativo.
- Se o aplicativo estiver usando o comando **TRANSFORM XMLTODATA**, o CICS emitirá uma mensagem e retornará uma resposta de solicitação inválida com um código RESP2 igual a 3 para o aplicativo.

Quando o CICS produz uma mensagem SOAP baseada no conteúdo de uma COMMAREA ou um contêiner, o atributo é transformado e incluído na mensagem. Quando um aplicativo usa o comando **TRANSFORM DATATOXML**, o CICS também transforma o atributo e o inclui no XML.

O Atributo que Permite Nil

O atributo nillable é um atributo especial que pode aparecer em um `xsd:element` em um esquema XML. Ele especifica que o atributo `xsi:nil` é válido para o elemento no XML. Se um elemento tiver o atributo `xsi:nil` especificado, isso indica que o elemento está presente, mas não tem valor e, portanto, nenhum conteúdo está associado a ele.

Se um esquema XML definiu o atributo nillable como true, ele é mapeado como um atributo necessário que utiliza um valor Booleano.

Quando o CICS recebe uma mensagem SOAP ou precisa transformar XML para um aplicativo que contém um atributo `xsi:nil`, o valor do atributo é true ou false. Se o valor for true, o aplicativo deverá ignorar os valores do elemento ou elementos aninhados no escopo do atributo `xsi:nil`.

Quando o CICS produz uma mensagem SOAP ou XML com base no conteúdo de uma COMMAREA ou um contêiner para o qual o valor para o atributo `xsi:nil` é true, ocorre o seguinte processamento:

- O atributo `xsi:nil` é gerado no XML ou mensagem SOAP.
- O valor do elemento associado é ignorado.
- Quaisquer elementos aninhados no elemento são ignorados.

Exemplo de Mensagem SOAP

Considere o esquema XML de exemplo a seguir, que poderia ser parte de um 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>
```

Aqui está um exemplo de uma mensagem SOAP parcial que está em conformidade com este esquema:

```
<root xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <num xsi:nil="true"/>
  <num>15</num>
  <num xsi:nil="true"/>
</root>
```

No COBOL, esta mensagem SOAP mapeia para estes 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)
```

Suporte para <xsd:any> e xsd:anyType:

DFHWS2LS e DFHSC2LS suportam o uso de <xsd:any> e xsd:anyType no esquema XML. É possível usar o elemento do esquema XML <xsd:any> para descrever uma seção de um documento XML com conteúdo indefinido. xsd:anyType é o tipo de dados de base a partir dos quais todos os tipos de dados simples e complexos são derivados; ele não tem restrições no conteúdo dos dados.

Antes de poder usar <xsd:any> e xsd:anyType com os assistentes CICS, configure os parâmetros a seguir:

- Configure o parâmetro **MAPPING-LEVEL** como 2.1 ou superior.
- Para um aplicativo do provedor de serviços da web, configure o parâmetro **PGMINT** como CHANNEL.

Exemplo de <xsd:any>

Este exemplo usa um elemento <xsd:any> para descrever algum conteúdo XML opcional não estruturado após a tag "Surname" na tag "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>
```

Uma mensagem SOAP de exemplo que está em conformidade com este esquema XML é:

```
<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>
```

Se esta mensagem SOAP é enviada ao CICS, o CICS preenche o contêiner Customer-xml-cont com os dados XML a seguir:

```
<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>
```

O CICS também preenche o contêiner Customer-xmlns-cont com as seguintes declarações de namespace XML que estão no escopo; essas declarações são separadas por um espaço:

```
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns="http://www.example.org/anyExample"
```

Exemplo de xsd:anyType

O xsd:anyType é o tipo de dados de base a partir do qual todos os tipos de dados simples e complexos são derivados. Ele não restringe o conteúdo de dados. Se você não especificar um tipo de dados, ele será padronizado como xsd:anyType; por exemplo, esses dois fragmentos de XML são equivalentes:

```
<xsd:element name="Name" type="xsd:anyType"/>
<xsd:element name="Name"/>
```

Estruturas de Linguagem Geradas

As estruturas de linguagem geradas para <xsd:any> ou xsd:anyType têm o formato a seguir em COBOL e um formato equivalente para as outras linguagens:

elementName-xml-cont PIC X(16)

O nome de um contêiner que contém o XML bruto. Quando o CICS processa uma mensagem SOAP recebida, ele coloca o subconjunto da mensagem SOAP que o <xsd:any> ou o xsd:anyType define neste contêiner. O aplicativo pode processar os dados XML somente nativamente. O aplicativo deve gerar o XML, preencher este contêiner e fornecer o nome do contêiner.

Este contêiner deve ser preenchido no modo de texto. Se o CICS preenche esse contêiner, ele faz isso usando a mesma variante de EBCDIC que o serviço da web está definido para usar. Caracteres que não existem na página de códigos EBCDIC de destino são substituídos por caracteres substitutos, mesmo se o contêiner é lido pelo aplicativo em UTF-8.

elementName+xmlns-cont PIC X(16)

O nome de um contêiner que contém quaisquer declarações de prefixo de namespace que estão no escopo. O conteúdo deste contêiner é semelhante àqueles do contêiner DFHWS-XMLNS, exceto que ele inclui todas as declarações de namespace que estão no escopo e que são relevantes, em vez de somente o subconjunto da tag SOAP Envelope.

Este contêiner deve ser preenchido no modo de texto. Se o CICS preenche esse contêiner, ele faz isso usando a mesma variante de EBCDIC que o serviço da web está definido para usar. Caracteres que não existem na página de códigos EBCDIC de destino são substituídos por caracteres substitutos, mesmo se o contêiner é lido pelo aplicativo em UTF-8.

Este contêiner é usado somente ao processar mensagens SOAP enviadas ao CICS. Se o aplicativo tentar fornecer um contêiner com declarações de namespace quando uma mensagem SOAP de saída for gerada, o contêiner e seu conteúdo serão ignorados pelo CICS. O CICS requer que o XML fornecido pelo aplicativo seja totalmente autocontido com relação a declarações de namespace.

O nome do elemento XML que contém o elemento `<xsd:any>` é incluído nos nomes de variáveis que são gerados para o elemento `<xsd:any>`. No exemplo de `<xsd:any>`, o elemento `<xsd:any>` é aninhado dentro do elemento `<xsd:element name="Customer">` e os nomes de variáveis que são gerados para o elemento `<xsd:any>` são `Customer-xml-cont PIC X(16)` e `Customer+xmlns-cont PIC X(16)`.

Para um tipo `xsd:anyType`, o nome do elemento XML direto é usado; no exemplo de `xsd:anyType` acima, os nomes de variáveis são `Name-xml-cont PIC X(16)` e `Name+xmlns-cont PIC X(16)`.

Suporte para `<xsd:choice>`:

Um elemento `<xsd:choice>` indica que somente uma das opções no elemento pode ser utilizada. Os assistentes CICS fornecem graus variados de suporte para elementos `<xsd:choice>` nos vários níveis de mapeamento.

Suporte para `<xsd:choice>` no nível de mapeamento 2.2 e superior

No nível de mapeamento 2.2 e superior, DFHWS2LS e DFHSC2LS fornecem suporte aprimorado para elementos `<xsd:choice>`. Os assistentes geram um novo contêiner que armazena o valor associado ao elemento `<xsd:choice>`. Os assistentes geram estruturas de linguagem contendo o nome de um novo contêiner e um campo extra:

fieldname -enum

O campo discriminador para indicar qual das opções o elemento `<xsd:choice>` usará.

fieldname -cont

O nome do contêiner que armazena a opção a ser usada. Uma estrutura de linguagem adicional é gerada para mapear o valor da opção.

O fragmento de esquema XML a seguir inclui um 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>

```

Se este fragmento de esquema XML for processado no nível de mapeamento 2.2 ou superior, o assistente gerará as estruturas de linguagem COBOL a seguir:

```

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.

```

Limitações para <xsd:choice> no nível de mapeamento 2.2 e superior

DFHSC2LS e DFHWS2LS não suportam elementos <xsd:choice> aninhados; por exemplo, o XML a seguir não é suportado:

```

<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 e DFHWS2LS não suportam elementos <xsd:choice> recorrentes; por exemplo, o XML a seguir não é suportado:

```

<xsd:choice maxOccurs="2">
  <xsd:element name="name1" type="string"/>
</xsd:choice>

```

DFHSC2LS e DFHWS2LS suportam, no máximo, 255 opções em um elemento <xsd:choice>.

Suporte para <xsd:choice> no nível de mapeamento 2.1 e abaixo

No nível de mapeamento 2.1 e abaixo, o DFHWS2LS fornece suporte limitado para elementos <xsd:choice>. O DFHWS2LS trata cada uma das opções em um elemento <xsd:choice> como se fossem um elemento <xsd:sequence> que pode ocorrer no máximo uma vez.

Apenas uma das opções em um elemento <xsd:choice> pode ser usada, portanto, tenha cuidado ao implementar um aplicativo usando o elemento <xsd:choice>

gerado somente com combinações válidas de opções. Cada um dos elementos possui seu próprio campo count na estrutura de linguagens gerada, exatamente um dos quais deve ser configurado como 1 e todos os outros devem ser configurados como 0. Qualquer outra combinação de valores é incorreta, exceto quando o <xsd:choice> é em si opcional, nesse caso é válido que todos os campos sejam configurados como 0.

Suporte para Grupos de Substituição:

É possível usar um grupo de substituição para definir um grupo de elementos XML que são intercambiáveis. Os assistentes do CICS fornecem suporte para grupos de substituição no nível de mapeamento 2.2 e superior.

No nível de mapeamento 2.2 e superior, DFHSC2LS e DFHWS2LS suportam grupos de substituição usando mapeamentos semelhantes àqueles usados para elementos <xsd:choice>. O assistente gera um campo de enumeração e um novo nome de contêiner na estrutura de linguagem.

O fragmento de esquema XML a seguir inclui uma matriz de dois elementos subGroupParent, cada um dos quais pode ser substituído por replacementOption1 ou 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" />
```

Processar este fragmento XML com o assistente gera as estruturas de linguagem COBOL a seguir:

```
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 obter mais informações sobre grupos de substituição, consulte a *especificação Esquema XML W3C Parte 1: Estruturas Segunda Edição*: http://www.w3.org/TR/xmlschema-1/#Elements_Equivalence_Class

Suporte para Elementos Abstratos e Tipos de Dados Abstratos:

Os assistentes do CICS fornecem suporte para elementos abstratos e tipos de dados abstratos no nível de mapeamento 2.2 e superior. Os assistentes do CICS mapeiam elementos abstratos e tipos de dados abstratos de uma maneira semelhante a grupos de substituição.

Suporte para elementos abstratos no nível de mapeamento 2.2 e superior

No nível de mapeamento 2.2 e superior, DFHSC2LS e DFHWS2LS tratam elementos abstratos quase da mesma maneira que grupos de substituição, exceto que o elemento abstrato não é um membro válido do grupo. Se não houver elementos substituíveis, o elemento abstrato será tratado como um elemento `<xsd:any>` e usará os mesmos mapeamentos que um elemento `<xsd:any>` no nível de mapeamento 2.1.

O fragmento de esquema XML a seguir especifica duas opções que podem ser usadas no lugar do elemento abstrato. O elemento abstrato em si não é uma opção 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" />
```

Processar este fragmento XML com o assistente gera as estruturas de linguagem COBOL a seguir:

```
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 obter mais informações sobre elementos abstratos, consulte a *especificação Esquema XML W3C Parte 0: Manual Segunda Edição*: <http://www.w3.org/TR/xmlschema-0/#SubsGroups>

Suporte para tipos de dados abstratos no nível de mapeamento 2.2 e superior

No nível de mapeamento 2.2 e superior, DFHSC2LS e DFHWS2LS tratam tipos de dados abstratos como grupos de substituição. O assistente gera um campo de enumeração e um novo nome de contêiner na estrutura de linguagem.

O fragmento de esquema XML a seguir especifica duas alternativas que podem ser usadas no lugar do tipo abstrato:

```
<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>
```

Processar este fragmento XML com o assistente gera as estruturas de linguagem COBOL a seguir:

```
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).
```

As estruturas de linguagem são geradas em copybooks separados. A estrutura de linguagem gerada para option1 é gerada em um copybook:

```
03 option1 PIC X(5).
```

A estrutura de linguagem para option2 é gerada em um copybook diferente:

```
03 option2 PIC X(10).
```

Para obter mais informações sobre tipos de dados abstratos, consulte a *especificação W3C XML Schema Parte 0: Primer Second Edition* : <http://www.w3.org/TR/xmlschema-0/#SubsGroups>

Como Manipular Conteúdo Variavelmente Repetido no COBOL:

No COBOL, não é possível processar conteúdo variavelmente repetido usando aritmética do ponteiro para endereçar cada instância dos dados. Outras linguagens de programação não possuem esta limitação. Este exemplo mostra como manipular o conteúdo de repetição variável no COBOL para um aplicativo de serviço da web.

Esta técnica também se aplica na transformação do XML em dados do aplicativo usando os comandos da API **TRANSFORM**. O documento WSDL de exemplo a seguir

representa um serviço da web com dados do aplicativo que consistem em uma sequência de 8 caracteres que ocorre um número variável de vezes:

```
<?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>
```

O processamento deste documento WSDL por meio do DFHWS2LS gera as estruturas de linguagem COBOL a seguir:

```
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).
```

Observe que o campo component de 8 caracteres é definido em uma estrutura separada chamada DFHWS-component. A estrutura de dados principal é chamada de applicationData e contém dois campos, component-num e component-cont. O campo

component-num indica quantas instâncias dos dados component estão presentes e o campo component-cont indica o nome de um contêiner que contém a lista concatenada de campos component.

O código COBOL a seguir demonstra uma maneira de processar a lista de dados variavelmente recorrentes. Ele faz uso de uma matriz de seção de ligação para endereçar instâncias subsequentes dos dados, cada uma das quais é exibida usando a instrução DISPLAY:

```
IDENTIFICATION DIVISION.  
PROGRAM-ID. EXVARY.
```

```
ENVIRONMENT DIVISION.  
DATA DIVISION.  
SEÇÃO DE ARMAZENAMENTO DE FUNCIONAMENTO.
```

```
* working storage variables  
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.
```

```
* a large linkage section array  
01 BIG-ARRAY PIC X(659999).  
  
* application data structures produced by DFHWS2LS  
* this is normally referenced with a COPY statement  
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.
```

```
* Get the DFHWS-DATA container  
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
```

```
* Get the recurring component data  
EXEC CICS GET CONTAINER(component-cont)  
SET(COMPONENT-PTR)  
FLENGTH(COMPONENT-DATA-LENGTH)  
END-EXEC
```

```
* Point the component structure at the first instance of the data  
SET ADDRESS OF DFHWS-component TO COMPONENT-PTR
```

```
* Store the length of a single component  
MOVE LENGTH OF DFHWS-component TO COMPONENT-LENGTH
```

```
* process each instance of component data in turn  
PERFORM WITH TEST AFTER
```

```

UNTIL COMPONENT-COUNT = component-num

* display the current instance of the data
DISPLAY 'component value is: ' component

* address the next instance of the component data
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

* end the loop
END-PERFORM.

* Point the component structure back at the first instance of
* of the data, for any further processing we may want to perform
SET ADDRESS OF DFHWS-component TO COMPONENT-PTR

* return to CICS.

EXEC CICS
RETURN
END-EXEC

GOBACK.

```

O código acima fornece uma solução genérica para manipular conteúdo de repetição variável. A matriz, BIG-ARRAY, move para o início de cada componente por vez e não permanece fixa no início dos dados. A estrutura de dados do componente é, então, movida para o ponto no primeiro byte do próximo componente. COMPONENT-PTR pode ser usado para recuperar a posição inicial dos dados do componente, se necessário.

Aqui está uma mensagem SOAP de exemplo que está em conformidade com o 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>

```

Aqui está a saída produzida pelo programa COBOL quando ele processa a mensagem SOAP:

```

CPIH 20080115103151 component value is: VALUE1
CPIH 20080115103151 component value is: VALUE2
CPIH 20080115103151 component value is: VALUE3

```

Matrizes de Elementos Variáveis

O XML pode conter uma matriz com números variados de elementos. Em geral, documentos WSDL e esquemas XML que contêm números variados de elementos não mapeiam eficientemente para uma única estrutura de dados de linguagem de alto nível. O CICS usa mapeamentos baseados em contêiner ou mapeamentos sequenciais para manipular números variados de elementos em XML.

Uma matriz com um número variado de elementos é representada no esquema XML usando os atributos minOccurs e maxOccurs na declaração de elemento:

- O atributo `minOccurs` especifica o número mínimo de vezes que o elemento pode ocorrer. Ela pode ter um valor 0 ou qualquer número inteiro positivo.
- O atributo `maxOccurs` especifica o número máximo de vezes que o elemento pode ocorrer. Ele pode ter um valor de qualquer número inteiro positivo maior ou igual ao valor do atributo `minOccurs`. Ele também pode ter um valor igual a `unbounded`, que indica que nenhum limite superior se aplica ao número de vezes que o elemento pode ocorrer.
- O valor padrão para ambos os atributos é 1.

Este exemplo denota uma sequência de 8 bytes que é opcional; ou seja, ela pode não ocorrer nunca ou ocorrer uma vez no XML do aplicativo ou na mensagem SOAP:

```
<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>
```

O exemplo a seguir denota uma sequência de 8 bytes que deve ocorrer pelo menos uma 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>
```

Em geral, documentos WSDL que contêm números variados de elementos não mapeiam eficientemente para uma única estrutura de dados de linguagem de alto nível. Portanto, para manipular esses casos, o CICS usa uma série de estruturas de dados conectadas que são transmitidas ao programa de aplicativo em uma série de contêineres. Essas estruturas são usadas como entrada e saída do aplicativo:

- Quando o CICS transforma XML em dados do aplicativo, ele preenche essas estruturas com os dados do aplicativo e o aplicativo os lê.
- Quando o CICS transforma os dados do aplicativo em XML, ele lê os dados do aplicativo nas estruturas que foram preenchidas pelo aplicativo.

O formato dessas estruturas de dados é explicado melhor com uma série de exemplos. O XML pode ser de uma mensagem SOAP ou de um aplicativo. Estes exemplos usam uma matriz de campos de 8 bytes simples. No entanto, o modelo suporta matrizes de tipos de dados complexos e matrizes de tipos de dados que contêm outras matrizes.

Número Fixo de Elementos

O primeiro exemplo ilustra um elemento que ocorre exatamente três vezes:

```
<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>

```

Neste exemplo, como o número de vezes que o elemento ocorre é conhecido antecipadamente, ele pode ser representado como uma matriz de comprimento fixo em uma declaração COBOL simples (ou o equivalente em outras linguagens):

```
05 component PIC X(8) OCCURS 3 TIMES
```

Número variado de elementos no nível de mapeamento 2 e abaixo

Este exemplo ilustra um elemento obrigatório que pode ocorrer de uma a cinco vezes:

```

<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>

```

A estrutura de dados principal contém uma declaração de dois campos. Quando o CICS transforma o XML em dados binários, o primeiro campo component-num contém o número de vezes que o elemento aparece no XML e o segundo campo, component-cont, contém o nome de um contêiner:

```
05 component-num PIC S9(9) COMP-5
05 component-cont PIC X(16)
```

Uma segunda estrutura de dados contém a declaração do elemento em si:

```
01 DFHWS-component
02 component PIC X(8)
```

Você deve examinar o valor de component-num (que conterá um valor no intervalo de 1 a 5) para descobrir quantas vezes o elemento ocorre. O conteúdo do elemento está no contêiner nomeado em component-cont; o contêiner contém uma matriz de elementos, em que cada elemento é mapeado pela estrutura de dados DFHWS-component.

Se minOccurs="0" e maxOccurs="1", o elemento é opcional. Para processar a estrutura de dados em seu programa de aplicativo, você deve examinar o valor de component-num:

- Se for zero, a mensagem não possui elemento do componente e o conteúdo de component-cont é indefinido.
- Se for um, o elemento do componente está no contêiner nomeado em component-cont.

O conteúdo do contêiner é mapeado pela estrutura de dados DFHWS-component.

Nota: Se a mensagem SOAP consistir em um único elemento recorrente, o DFHWS2LS gerará duas estruturas de linguagem. A estrutura de linguagem principal contém o número de elementos na matriz e o nome de um contêiner que contém a matriz de elementos. A segunda estrutura de linguagem mapeia uma única instância do elemento recorrente.

Número variado de elementos no nível de mapeamento 2.1 e superior

No nível de mapeamento 2.1 e acima, é possível usar o parâmetro **INLINE-MAXOCCURS-LIMIT** nos assistentes do CICS. O parâmetro **INLINE-MAXOCCURS-LIMIT** especifica a maneira como números variados de elementos são manipulados. As opções de mapeamento para números variados de elementos são mapeamento baseado em contêiner, descrito em “Número variado de elementos no nível de mapeamento 2 e abaixo” na página 348, ou mapeamento sequencial. O *value* deste parâmetro pode ser um número inteiro positivo no intervalo de 0 a 32767:

- O valor padrão de **INLINE-MAXOCCURS-LIMIT** é 1, o qual assegura que elementos opcionais sejam mapeados sequencialmente.
- Um valor 0 para o parâmetro **INLINE-MAXOCCURS-LIMIT** evita o mapeamento sequencial.
- Se maxOccurs for menor ou igual ao valor de **INLINE-MAXOCCURS-LIMIT**, o mapeamento sequencial será usado.
- Se maxOccurs for maior do que o valor de **INLINE-MAXOCCURS-LIMIT**, o mapeamento baseado em contêiner será usado.

O mapeamento de números variados de elementos sequenciais resulta na geração de uma matriz, como acontece com o exemplo de ocorrência fixa acima, e de um contador. O campo component-num indica quantas instâncias do elemento estão presentes e elas são apontadas pela matriz. Para o exemplo mostrado em “Número variado de elementos no nível de mapeamento 2 e abaixo” na página 348, quando **INLINE-MAXOCCURS-LIMIT** é menor ou igual a 5, a estrutura de dados gerada conforme a seguir:

```
05 component-num PIC S9(9) COMP-5 SYNC.  
05 component OCCURS 5 PIC X(8).
```

O primeiro campo, component-num, é idêntico ao de saída para o mapeamento baseado em contêiner de exemplo na seção anterior. O segundo campo contém uma matriz de comprimento 5 que é grande o suficiente para conter o número máximo de elementos que podem ser gerados.

O mapeamento sequencial difere do mapeamento baseado em contêiner, que armazena o número de ocorrências do elemento e o nome do contêiner no qual os dados são colocados, porque ele armazena todos os dados no contêiner atual. O armazenamento dos dados no contêiner atual geralmente irá melhorar o desempenho e tornar o mapeamento sequencial preferível.

Matrizes de Variáveis Aninhadas

Os documentos WSDL e esquemas XML complexos podem conter elementos variavelmente recorrentes, que por sua vez contêm os elementos variavelmente recorrentes. Nesse caso, a estrutura descrita se estende além dos dois níveis descritos nos exemplos.

Este exemplo ilustra um elemento opcional chamado <component2> que está aninhado em um elemento obrigatório chamado <component1>, em que o elemento obrigatório pode ocorrer de uma a cinco vezes:

```
<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>

```

A estrutura de dados de nível superior é exatamente a mesma que nos exemplos anteriores:

```

05 component1-num PIC S9(9) COMP-5
05 component1-cont PIC X(16)

```

No entanto, a segunda estrutura de dados contém estes elementos:

```

01 DFHWS-component1
02 component2-num PIC S9(9) COMP-5
02 component2-cont PIC X(16)

```

Uma estrutura de terceiro nível contém estes elementos:

```

01 DFHWS-component2
02 component2 PIC X(8)

```

O número de ocorrências do elemento <component1> mais externo está em component1-num.

O contêiner nomeado em component1-cont contém uma matriz com esse número de instâncias da segunda estrutura de dados DFHWS-component1.

Cada instância de component2-cont nomeia um contêiner diferente, cada um dos quais contém a estrutura de dados mapeada pela estrutura de terceiro nível DFHWS-component2.

Para ilustrar essa estrutura, considere o fragmento de XML que corresponde ao exemplo:

```

<component1><component2>
string1
</component2></component1>
<component1><component2>
string2
</component2></component1>
<component1></component1>

```

<component1> ocorre três vezes. Cada um dos dois primeiros contém uma instância de <component2>; a terceira instância não.

Na estrutura de dados de nível superior, component1-num contém um valor igual a 3. O contêiner nomeado em component1-cont possui três instâncias de DFHWS-component1 :

1. Na primeira, component2-num possui um valor 1 e o contêiner nomeado em component2-cont contém *string1*.
2. Na segunda, component2-num possui um valor 1 e o contêiner nomeado em component2-cont contém *string2*.
3. Na terceira, component2-num possui um valor 0 e o conteúdo de component2-cont é indefinido.

Neste exemplo, a estrutura de dados completa é representada por quatro contêineres:

- A estrutura de dados raiz no contêiner DFHWS-DATA
- O contêiner nomeado em component1-cont
- Dois contêineres nomeados nas duas primeiras instâncias de component2-cont

Estruturas Opcionais e `xsd:choice`

DFHWS2LS e DFHSC2LS suportam o uso de `maxOccurs` e `minOccurs` nos elementos `<xsd:sequence>`, `<xsd:choice>` e `<xsd:all>` somente no nível de mapeamento 2.1 e acima, em que os atributos `minOccurs` e `maxOccurs` são configurados como `minOccurs="0"` e `maxOccurs="1"`.

Os assistentes geram mapeamentos que tratam estes elementos como se cada elemento filho neles fosse opcional. Ao implementar um aplicativo com estes elementos, assegure que combinações inválidas de opções não sejam geradas pelo aplicativo. Cada um dos elementos possui seu próprio campo `count` na estrutura de linguagens gerada, todos estes campos devem ser configurados como "0" ou todos devem ser configurados como "1". Qualquer outra combinação de valores é inválida, exceto para elementos `<xsd:choice>`.

Elementos `<xsd:choice>` indicam que somente uma das opções no elemento pode ser usada. Isto é suportado em todos os níveis de mapeamento. Os assistentes manipulam cada uma das opções em um `<xsd:choice>` como se estivesse em um elemento `<xsd:sequence>` com `minOccurs="0"` e `maxOccurs="1"`. Tome cuidado ao implementar um aplicativo usando o elemento `<xsd:choice>` para assegurar que combinações inválidas de opções não sejam geradas pelo aplicativo. Cada um dos elementos possui seu próprio campo `count` na estrutura de linguagens gerada, exatamente um dos quais deve ser configurado como '1' e todos os outros devem ser configurados como '0'. Qualquer outra combinação de valores é inválida, exceto quando o elemento `<xsd:choice>` é em si opcional, nesse caso ele é válido para todos os campos a serem configurados como '0'.

Suporte para Valores de Comprimento Variável e Espaço em Branco

É possível customizar a maneira na qual os valores de comprimento variável e espaços em branco são manipulados usando as configurações nos assistentes CICS e incluindo máscaras diretamente no esquema XML.

Geralmente, o assistente CICS XML e o assistente de serviço da web do CICS mapeiam sequências de dados para matrizes de caracteres de comprimento fixo; essas matrizes requerem preenchimento com espaços ou nulos. O mapeamento de valores de comprimento variável para matrizes de dados de comprimento fixo pode ser ineficiente e desperdiçar armazenamento. Se o comprimento de seus dados é variável, é recomendável customizar a maneira como esses mapeamentos são manipulados.

Se você estiver convertendo a partir de uma estrutura de linguagem para um esquema XML ou documento WSDL, é recomendável especificar as máscaras `whiteSpace` e `maxLength` em seu esquema XML e configurar o parâmetro **CHAR-VARYING-LIMIT** nos assistentes.

Se você estiver convertendo a partir de um esquema XML ou documento WSDL para uma estrutura de linguagem, é recomendável configurar um valor apropriado para o parâmetro **CHAR-VARYING** nos assistentes.

Nota: Caracteres nulos ('x00') não são válidos em documentos XML. Quaisquer caracteres nulos de dados do aplicativo analisados pelo CICS são vistos como o fim de uma sequência e o valor é truncado. Quando o CICS gera dados do aplicativo, ele faz isso de acordo com o valor do parâmetro **CHAR-VARYING**. Por exemplo, se a opção **CHAR-VARYING=NULL** for especificada, as sequências de comprimento variável geradas pelo CICS serão finalizadas com um caractere nulo.

Mapeando valores de comprimento variável a partir do XML para estruturas de linguagem

Use máscaras no esquema XML ou especifique determinados parâmetros nos assistentes CICS para customizar a maneira na qual os mapeamentos entre o esquema XML ou o documento WSDL e a estrutura de linguagem são manipulados.

Tipos de dados XML podem ser restringidos usando máscaras. Use as máscaras de comprimento (`length`, `maxLength` e `minLength`) e a máscara `whiteSpace` para customizar a maneira como os dados de comprimento variável em seu XML são manipulados.

length

Usado para especificar que os dados são de comprimento fixo.

maxLength

Usado para especificar o comprimento máximo para o tipo de dados. Se esse valor não for configurado para um tipo de dados baseado em sequência, o comprimento máximo será ilimitado.

minLength

Usado para especificar o comprimento mínimo para o tipo de dados. Se esse valor não for configurado para um tipo de dados baseado em sequência, o comprimento mínimo será 0.

whiteSpace

Usado para especificar como o espaço em branco ao redor de um valor de dados é manipulado. O espaço em branco inclui espaços, tabulações e novas linhas. A máscara `whiteSpace` pode ser configurada como `preserve`, `replace` ou `collapse`:

- Um valor `preserve` mantém qualquer espaço em branco no valor dos dados.
- Um valor `replace` significa que quaisquer tabulações ou novas linhas serão substituídas pelo número apropriado de espaços.
- Um valor `collapse` significa que espaços em branco iniciais, finais e integrados são removidos e que todas as tabulações, novas linhas e espaços consecutivos são substituídos por caracteres de espaço único.

Se a máscara `whiteSpace` não for configurada, o espaço em branco será preservado.

Para obter mais informações sobre as máscaras de esquema XML, consulte o esquema de recomendação do W3C *Esquema XML Parte 2: Tipos de Dados Segunda Edição* <http://www.w3.org/TR/xmlschema-2/#facets>

Os parâmetros a seguir nos assistentes CICS, DFHSC2LS e DFHWS2LS, podem ser usados para alterar a maneira como os dados de comprimento variável são mapeados a partir do esquema XML para a estrutura de linguagem. Esses parâmetros estão disponíveis no nível de mapeamento 1.2 ou superior.

DEFAULT-CHAR-MAXLENGTH

Especifica o comprimento da matriz padrão de dados de caractere em caracteres para mapeamentos em que nenhum comprimento está implícito no esquema XML ou documento WSDL. O valor desse parâmetro pode ser um número inteiro positivo no intervalo de 1 a 2147483647.

No entanto, é recomendável especificar o comprimento máximo de caracteres que você deseja que DFHSC2LS ou DFHWS2LS use diretamente em seu esquema XML ou documento WSDL com a máscara `maxLength`. Especificar o comprimento máximo diretamente no esquema XML ou documento WSDL evita problemas associados a ter um padrão global aplicado a todos os tipos de dados baseados em sequência.

CHAR-VARYING-LIMIT

Especifica o tamanho máximo dos dados de caractere de comprimento variável que são mapeados para a estrutura de linguagem. Se os dados de caractere forem maiores que o valor especificado neste parâmetro, eles serão mapeados para um contêiner e o nome do contêiner será usado na estrutura de linguagem gerada. O valor pode variar de 0 até o padrão de 32767 bytes.

CHAR-VARYING

Especifica como os dados de caracteres de comprimento variável são mapeados. Se você não especificar esse parâmetro, o mapeamento padrão dependerá da linguagem especificada. Você pode selecionar estas opções:

- **CHAR-VARYING=NO** especifica que os dados de caracteres de comprimento variável são mapeados como sequências de comprimento fixo.
- **CHAR-VARYING=NULL** especifica que dados de caracteres de comprimento variável são mapeados para sequências com terminação nula.
- **CHAR-VARYING=YES** especifica que dados de caracteres de comprimento variável são mapeados para um tipo de dados CHAR VARYING em PL/I. Nas linguagens COBOL, C e C++, os dados de caracteres de comprimento variável são mapeados para uma representação equivalente que consiste em dois elementos relacionados: o comprimento dos dados e os dados.

A configuração de **CHAR-VARYING=YES** geralmente resulta no melhor desempenho.

Mapeando valores de comprimento variável a partir de estruturas de linguagem para XML

É possível customizar a maneira na qual os mapeamentos entre a estrutura de linguagem e o esquema XML, ou documento WSDL, são manipulados. Configure o parâmetro **CHAR-VARYING** em DFHLS2SC ou DFHLS2WS para **COLLAPSE** ou **NULL** para mudar a maneira como as matrizes de caracteres são geradas.

A configuração da opção **CHAR-VARYING=NULL** informa ao CICS para incluir um caractere nulo no final de cada matriz de caracteres ao gerar XML.

A configuração da opção **CHAR-VARYING=COLLAPSE** informa ao CICS para remover automaticamente quaisquer espaços à direita do final das matrizes de caracteres ao gerar o XML. Essa opção está disponível somente no nível de mapeamento 2.1 ou superior e **CHAR-VARYING=COLLAPSE** é o valor padrão no nível de mapeamento 2.1 ou superior para todas as linguagens diferentes de C e C++. Quando o XML é analisado, todos os espaços em branco iniciais, finais e integrados são removidos.

Para obter mais informações, consulte Suporte para espaço em branco e valores de comprimento variável nos serviços da web do CICS (nota técnica) .

Matrizes variáveis de elementos no DFHJS2LS

O JSON pode conter matrizes de números variados de elementos. Em geral, Esquemas JSON que contêm números variados de elementos não mapeiam eficientemente para uma única estrutura de dados de linguagem de alto nível. O CICS usa mapeamentos baseados em contêiner ou mapeamentos sequenciais para manipular números variados de elementos em dados JSON.

Uma matriz com um número variado de elementos é representada no esquema JSON usando as palavras-chave `minItems` e `maxItems` no esquema com valor `"type"` igual a `"array"` :

- A palavra-chave `minItems` especifica o número mínimo de vezes que o elemento pode ocorrer. Ela pode ter um valor 0 ou qualquer número inteiro positivo. Ela é padronizada com o valor 0.
- A palavra-chave `maxItems` especifica o número máximo de vezes que o elemento pode ocorrer. Ele pode ter um valor de qualquer número inteiro positivo maior ou igual ao valor da palavra-chave `minItems`.
- Se a palavra-chave `maxItems` está ausente, significa que a matriz é ilimitada.

Um campo opcional pode ser indicado por uma matriz variável de `"maxItems":1`. Por exemplo, uma sequência opcional 8 bytes chamada `"component"` :

```
"properties":{
  "component": {
    "type":"array",
    "maxItems":1,
    "items":{
      "type" : "string",
      "maxLength": 8
    }
  },
  "required": ["component"]
}
```

O mesmo efeito pode ser produzido não incluindo o nome do campo no valor da palavra-chave `"required"`:

```
"properties":{
  "component": {
    "type" : "string",
    "maxLength": 8
  }
}
```

Em geral, esquemas JSON que contêm números variados de elementos não mapeiam eficientemente para uma única estrutura de dados de linguagem de alto nível. Para lidar com esses casos, o CICS usa uma série de estruturas de dados conectadas que são transmitidas ao programa de aplicativo em uma série de contêineres. Essas estruturas são usadas como entrada e saída do aplicativo:

- Quando o CICS transforma dados JSON em dados do aplicativo, ele preenche estas estruturas com os dados do aplicativo e o aplicativo os lê.
- Quando o CICS transforma os dados do aplicativo em dados JSON, ele lê os dados do aplicativo nas estruturas que foram preenchidas pelo aplicativo.

Os exemplos a seguir ilustram o formato dessas estruturas de dados. Estes exemplos usam uma matriz de campos de 8 bytes simples. No entanto, o modelo suporta matrizes de tipos de dados complexos e matrizes de tipos de dados que contêm outras matrizes.

Exemplo 1. Número fixo de elementos

Este exemplo ilustra um elemento que ocorre exatamente três vezes:

```
"properties":{
  "component": {
    "type": "array",
    "maxItems": 3,
    "minItems": 3,
    "items":{
      "type" : "string",
      "maxLength": 8
    }
  },
  "required": ["component"]
```

Neste exemplo, o número de vezes que o elemento ocorre é conhecido antecipadamente, portanto, ele pode ser representado como uma matriz de comprimento fixo em uma declaração COBOL simples ou o equivalente em outras linguagens.

```
05 component PIC X(8) OCCURS 3 TIMES
```

Exemplo 2. Ativando o número de elementos no nível de mapeamento 2 e abaixo

Este exemplo ilustra um elemento obrigatório que pode ocorrer de uma a cinco vezes:

```
"properties":{
  "component": {
    "type": "array",
    "maxItems": 5,
    "minItems": 1,
    "items":{
      "type" : "string",
      "maxLength": 8
    }
  },
  "required": ["component"]
```

A estrutura de dados principal contém uma declaração de dois campos. Quando o CICS transforma os dados JSON em dados binários, o primeiro campo component-num contém o número de vezes que o elemento aparece nos dados JSON e o segundo campo, component-cont, contém o nome de um contêiner:

```
05 component-num PIC S9(9) COMP-5
05 component-cont PIC X(16)
```

Uma segunda estrutura de dados contém a declaração do elemento em si:

```
01 DFHJS-component
02 component PIC X(8)
```

Você deve examinar o valor de component-num, que conterá um valor no intervalo de 1 a 5, para descobrir quantas vezes o elemento ocorre. O conteúdo do elemento

está no contêiner nomeado em `component-cont`; o contêiner contém uma matriz de elementos, em que cada elemento é mapeado pela estrutura de dados `DFHJS-component`.

Se `minItems="0"`, ou está ausente, e `maxItems="1"`, o elemento é opcional. Para processar a estrutura de dados em seu programa de aplicativo, você deve examinar o valor de `component-num`:

- Se for zero, a mensagem não possui elemento do componente e o conteúdo de `component-cont` é indefinido.
- Se for um, o elemento do componente está no contêiner nomeado em `component-cont`.

O conteúdo do contêiner é mapeado pela estrutura de dados `DFHJS-component`.

Nota: Se os dados JSON consistirem em um único elemento recorrente, o `DFHJS2LS` gerará duas estruturas de linguagem. A estrutura de linguagem principal contém o número de elementos na matriz e o nome de um contêiner que contém a matriz de elementos. A segunda estrutura de linguagem mapeia uma única instância do elemento recorrente.

Exemplo 3. Número variado de elementos no nível de mapeamento 2.1 e superior

No nível de mapeamento 2.1 e acima, é possível usar o parâmetro **INLINE-MAXOCCURS-LIMIT** nos assistentes do CICS. O parâmetro **INLINE-MAXOCCURS-LIMIT** especifica a maneira como números variados de elementos são manipulados. As opções de mapeamento para números variados de elementos são o mapeamento baseado em contêiner, descrito em “Exemplo 2. Ativando o número de elementos no nível de mapeamento 2 e abaixo” na página 283 ou o mapeamento sequencial. O valor desse parâmetro pode ser um número inteiro positivo no intervalo de 0 a 32767:

- O valor padrão de **INLINE-MAXOCCURS-LIMIT** é 1, o qual assegura que elementos opcionais sejam mapeados sequencialmente.
- Um valor 0 para o parâmetro **INLINE-MAXOCCURS-LIMIT** evita o mapeamento sequencial.
- Se `maxItems` for menor ou igual ao valor de **INLINE-MAXOCCURS-LIMIT**, o mapeamento sequencial será usado.
- Se `maxItems` for maior do que o valor de **INLINE-MAXOCCURS-LIMIT**, o mapeamento baseado em contêiner será usado.

O mapeamento de números variados de elementos sequenciais resulta na geração de uma matriz, como acontece com o exemplo de ocorrência fixa acima, e de um contador. O campo `component-num` indica quantas instâncias do elemento estão presentes e elas são apontadas pela matriz. Para o exemplo mostrado em “Exemplo 2. Ativando o número de elementos no nível de mapeamento 2 e abaixo” na página 283, quando **INLINE-MAXOCCURS-LIMIT** é menor ou igual a 5, a estrutura de dados gerada conforme a seguir:

```
05 component-num PIC S9(9) COMP-5 SYNC.  
05 component OCCURS 5 PIC X(8).
```

O primeiro campo, `component-num`, é idêntico ao de saída para o mapeamento baseado em contêiner de exemplo na seção anterior. O segundo campo contém uma matriz de comprimento 5 que é grande o suficiente para conter o número máximo de elementos que podem ser gerados.

O mapeamento sequencial difere do mapeamento baseado em contêiner, que armazena o número de ocorrências do elemento e o nome do contêiner no qual os dados são colocados, porque ele armazena todos os dados no contêiner atual. O armazenamento dos dados no contêiner atual geralmente irá melhorar o desempenho e tornar o mapeamento sequencial preferível.

Exemplo 4. Matrizes variáveis aninhadas

Os esquemas JSON complexos podem conter elementos variáveis recorrentes, que que por sua vez contêm elementos variáveis recorrentes. Nesse caso, a estrutura descrita se estende além dos dois níveis descritos nos exemplos.

Este exemplo ilustra um elemento opcional chamado "component2" que está aninhado em um elemento obrigatório chamado "component1", onde o elemento obrigatório pode ocorrer de uma a cinco vezes:

```
"properties":{
  "component1": {
    "type": "array",
    "maxItems": 5,
    "minItems": 1,
    "items":{
      "type": "object",
      "properties":{
        "component2":{
          "type" : "string",
          "maxLength": 8
        }
      },
      "required": ["component2"]
    }
  },
  "required": ["component1"]
}
```

A estrutura de dados de nível superior é exatamente a mesma que nos exemplos anteriores:

```
05 component1-num PIC S9(9) COMP-5
05 component1-cont PIC X(16)
```

No entanto, a segunda estrutura de dados contém estes elementos:

```
01 DFHJS-component1
02 component2-num PIC S9(9) COMP-5
02 component2-cont PIC X(16)
```

Uma estrutura de terceiro nível contém estes elementos:

```
01 DFHJS-component2
02 component2 PIC X(8)
```

O número de ocorrências do elemento "component1" mais externo está em component1-num.

O contêiner nomeado em component1-cont contém uma matriz com esse número de instâncias da segunda estrutura de dados DFHJS-component1.

Cada instância de component2-cont nomeia um contêiner diferente, cada um dos quais contém a estrutura de dados mapeada pela estrutura de terceiro nível DFHJS-component2.

Para ilustrar essa estrutura, considere o fragmento de dados JSON que corresponde ao exemplo:

```

{"component1":
[
{
"component2": "string1"
},
{
"component2": "string2"
},
]
}

```

"component1" ocorre três vezes. Os dois primeiros contêm uma instância de "component2", a terceira instância não.

Na estrutura de dados de nível superior, component1-num contém um valor igual a 3. O contêiner nomeado em component1-cont possui três instâncias de DFHJS-component1 :

1. Na primeira, component2-num possui um valor 1 e o contêiner nomeado em component2-cont contém *string1*.
2. Na segunda, component2-num possui um valor 1 e o contêiner nomeado em component2-cont contém *string2*.
3. Na terceira, component2-num possui um valor 0 e o conteúdo de component2-cont é indefinido.

Neste exemplo, a estrutura de dados completa é representada por quatro contêineres:

- A estrutura de dados raiz no contêiner DFHJS-DATA.
- O contêiner nomeado em component1-cont.
- Dois contêineres nomeados nas duas primeiras instâncias de component2-cont.

Estruturas opcionais e a palavra-chave **required**

Estruturas de dados são definidas pelo Esquema JSON "type" de "object". Os esquemas relacionam nomes de campos a tipos individuais usando o objeto fornecido pela palavra-chave "properties". O requisito para esses campos fazerem parte dos dados JSON descritos pelo Esquema JSON é controlado pela matriz fornecida pela palavra-chave "required". Essa matriz lista todos os nomes de campos que devem estar presentes nos dados JSON. Os campos opcionais são, portanto, representados por sua ausência nessa matriz ou, como a matriz não tem permissão para estar vazia, a ausência da palavra-chave "required". Nesse caso, todos os campos são opcionais.

Os campos opcionais são tratados como uma matriz variável de 0 ou 1 elemento. Isso inclui um campo adicional com o sufixo "-num" anexado ao nome de elemento. Se o comprimento total é maior que 28 caracteres, o nome de elemento é truncado. No tempo de execução isso será diferente de zero para indicar que o valor estava presente nos dados JSON e zero se não estava.

Este exemplo mostra dois campos, um necessário chamado "required-structure" e o outro opcional chamado "optional-structure" :

```

{
"required-structure": {
"properties": {
"string": {
"required-structure": {
"type": "string",
"maxLength": 8
},

```

```

"optional-structure": {
  "type": "string",
  "maxLength": 8
},
"required": [
  "required-structure"
]
}

```

A estrutura COBOL gerada mostra ambos os campos, mas o segundo é precedido por "optional-structure-num" que é uma contagem de número inteiro dos elementos, com 0 representando nenhum e 1 indicando que ele está presente. O valor é configurado para indicar se o "optional-structure" contém dados válidos ou não.

```

03 OutputData.
06 required-structure PIC X(8).
06 optional-structure-num PIC S9(9) COMP-5 SYNC.
06 optional-structure PIC X(8).

```

Matrizes de Elementos Variáveis

O XML pode conter uma matriz com números variados de elementos. Em geral, documentos WSDL e esquemas XML que contêm números variados de elementos não mapeiam eficientemente para uma única estrutura de dados de linguagem de alto nível. O CICS usa mapeamentos baseados em contêiner ou mapeamentos sequenciais para manipular números variados de elementos em XML.

Uma matriz com um número variado de elementos é representada no esquema XML usando os atributos `minOccurs` e `maxOccurs` na declaração de elemento:

- O atributo `minOccurs` especifica o número mínimo de vezes que o elemento pode ocorrer. Ela pode ter um valor 0 ou qualquer número inteiro positivo.
- O atributo `maxOccurs` especifica o número máximo de vezes que o elemento pode ocorrer. Ele pode ter um valor de qualquer número inteiro positivo maior ou igual ao valor do atributo `minOccurs`. Ele também pode ter um valor igual a `unbounded`, que indica que nenhum limite superior se aplica ao número de vezes que o elemento pode ocorrer.
- O valor padrão para ambos os atributos é 1.

Este exemplo denota uma sequência de 8 bytes que é opcional; ou seja, ela pode não ocorrer nunca ou ocorrer uma vez no XML do aplicativo ou na mensagem SOAP:

```

<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>

```

O exemplo a seguir denota uma sequência de 8 bytes que deve ocorrer pelo menos uma 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>

```

Em geral, documentos WSDL que contêm números variados de elementos não mapeiam eficientemente para uma única estrutura de dados de linguagem de alto nível. Portanto, para manipular esses casos, o CICS usa uma série de estruturas de dados conectadas que são transmitidas ao programa de aplicativo em uma série de contêineres. Essas estruturas são usadas como entrada e saída do aplicativo:

- Quando o CICS transforma XML em dados do aplicativo, ele preenche essas estruturas com os dados do aplicativo e o aplicativo os lê.
- Quando o CICS transforma os dados do aplicativo em XML, ele lê os dados do aplicativo nas estruturas que foram preenchidas pelo aplicativo.

O formato dessas estruturas de dados é explicado melhor com uma série de exemplos. O XML pode ser de uma mensagem SOAP ou de um aplicativo. Estes exemplos usam uma matriz de campos de 8 bytes simples. No entanto, o modelo suporta matrizes de tipos de dados complexos e matrizes de tipos de dados que contêm outras matrizes.

Número Fixo de Elementos

O primeiro exemplo ilustra um elemento que ocorre exatamente três vezes:

```

<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>

```

Neste exemplo, como o número de vezes que o elemento ocorre é conhecido antecipadamente, ele pode ser representado como uma matriz de comprimento fixo em uma declaração COBOL simples (ou o equivalente em outras linguagens):

```
05 component PIC X(8) OCCURS 3 TIMES
```

Número variado de elementos no nível de mapeamento 2 e abaixo

Este exemplo ilustra um elemento obrigatório que pode ocorrer de uma a cinco vezes:

```

<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>

```

A estrutura de dados principal contém uma declaração de dois campos. Quando o CICS transforma o XML em dados binários, o primeiro campo component-num contém o número de vezes que o elemento aparece no XML e o segundo campo, component-cont, contém o nome de um contêiner:

```
05 component-num PIC S9(9) COMP-5
05 component-cont PIC X(16)
```

Uma segunda estrutura de dados contém a declaração do elemento em si:

```
01 DFHWS-component  
02 component PIC X(8)
```

Você deve examinar o valor de `component-num` (que conterá um valor no intervalo de 1 a 5) para descobrir quantas vezes o elemento ocorre. O conteúdo do elemento está no contêiner nomeado em `component-cont`; o contêiner contém uma matriz de elementos, em que cada elemento é mapeado pela estrutura de dados `DFHWS-component`.

Se `minOccurs="0"` e `maxOccurs="1"`, o elemento é opcional. Para processar a estrutura de dados em seu programa de aplicativo, você deve examinar o valor de `component-num`:

- Se for zero, a mensagem não possui elemento do componente e o conteúdo de `component-cont` é indefinido.
- Se for um, o elemento do componente está no contêiner nomeado em `component-cont`.

O conteúdo do contêiner é mapeado pela estrutura de dados `DFHWS-component`.

Nota: Se a mensagem SOAP consistir em um único elemento recorrente, o `DFHWS2LS` gerará duas estruturas de linguagem. A estrutura de linguagem principal contém o número de elementos na matriz e o nome de um contêiner que contém a matriz de elementos. A segunda estrutura de linguagem mapeia uma única instância do elemento recorrente.

Número variado de elementos no nível de mapeamento 2.1 e superior

No nível de mapeamento 2.1 e acima, é possível usar o parâmetro **INLINE-MAXOCCURS-LIMIT** nos assistentes do CICS. O parâmetro **INLINE-MAXOCCURS-LIMIT** especifica a maneira como números variados de elementos são manipulados. As opções de mapeamento para números variados de elementos são mapeamento baseado em contêiner, descrito em “Número variado de elementos no nível de mapeamento 2 e abaixo” na página 348, ou mapeamento sequencial. O *value* deste parâmetro pode ser um número inteiro positivo no intervalo de 0 a 32767:

- O valor padrão de **INLINE-MAXOCCURS-LIMIT** é 1, o qual assegura que elementos opcionais sejam mapeados sequencialmente.
- Um valor 0 para o parâmetro **INLINE-MAXOCCURS-LIMIT** evita o mapeamento sequencial.
- Se `maxOccurs` for menor ou igual ao valor de **INLINE-MAXOCCURS-LIMIT**, o mapeamento sequencial será usado.
- Se `maxOccurs` for maior do que o valor de **INLINE-MAXOCCURS-LIMIT**, o mapeamento baseado em contêiner será usado.

O mapeamento de números variados de elementos sequenciais resulta na geração de uma matriz, como acontece com o exemplo de ocorrência fixa acima, e de um contador. O campo `component-num` indica quantas instâncias do elemento estão presentes e elas são apontadas pela matriz. Para o exemplo mostrado em “Número variado de elementos no nível de mapeamento 2 e abaixo” na página 348, quando **INLINE-MAXOCCURS-LIMIT** é menor ou igual a 5, a estrutura de dados gerada conforme a seguir:

```
05 component-num PIC S9(9) COMP-5 SYNC.  
05 component OCCURS 5 PIC X(8).
```

O primeiro campo, `component-num`, é idêntico ao de saída para o mapeamento baseado em contêiner de exemplo na seção anterior. O segundo campo contém uma matriz de comprimento 5 que é grande o suficiente para conter o número máximo de elementos que podem ser gerados.

O mapeamento sequencial difere do mapeamento baseado em contêiner, que armazena o número de ocorrências do elemento e o nome do contêiner no qual os dados são colocados, porque ele armazena todos os dados no contêiner atual. O armazenamento dos dados no contêiner atual geralmente irá melhorar o desempenho e tornar o mapeamento sequencial preferível.

Matrizes de Variáveis Aninhadas

Os documentos WSDL e esquemas XML complexos podem conter elementos variavelmente recorrentes, que por sua vez contêm os elementos variavelmente recorrentes. Nesse caso, a estrutura descrita se estende além dos dois níveis descritos nos exemplos.

Este exemplo ilustra um elemento opcional chamado `<component2>` que está aninhado em um elemento obrigatório chamado `<component1>`, em que o elemento obrigatório pode ocorrer de uma a cinco vezes:

```
<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>
```

A estrutura de dados de nível superior é exatamente a mesma que nos exemplos anteriores:

```
05 component1-num PIC S9(9) COMP-5
05 component1-cont PIC X(16)
```

No entanto, a segunda estrutura de dados contém estes elementos:

```
01 DFHWS-component1
02 component2-num PIC S9(9) COMP-5
02 component2-cont PIC X(16)
```

Uma estrutura de terceiro nível contém estes elementos:

```
01 DFHWS-component2
02 component2 PIC X(8)
```

O número de ocorrências do elemento `<component1>` mais externo está em `component1-num`.

O contêiner nomeado em `component1-cont` contém uma matriz com esse número de instâncias da segunda estrutura de dados `DFHWS-component1`.

Cada instância de component2-cont nomeia um contêiner diferente, cada um dos quais contém a estrutura de dados mapeada pela estrutura de terceiro nível DFHWS-component2.

Para ilustrar essa estrutura, considere o fragmento de XML que corresponde ao exemplo:

```
<component1><component2>
string1
</component2></component1>
<component1><component2>
string2
</component2></component1>
<component1></component1>
```

<component1> ocorre três vezes. Cada um dos dois primeiros contém uma instância de <component2>; a terceira instância não.

Na estrutura de dados de nível superior, component1-num contém um valor igual a 3. O contêiner nomeado em component1-cont possui três instâncias de DFHWS-component1 :

1. Na primeira, component2-num possui um valor 1 e o contêiner nomeado em component2-cont contém *string1*.
2. Na segunda, component2-num possui um valor 1 e o contêiner nomeado em component2-cont contém *string2*.
3. Na terceira, component2-num possui um valor 0 e o conteúdo de component2-cont é indefinido.

Neste exemplo, a estrutura de dados completa é representada por quatro contêineres:

- A estrutura de dados raiz no contêiner DFHWS-DATA
- O contêiner nomeado em component1-cont
- Dois contêineres nomeados nas duas primeiras instâncias de component2-cont

Estruturas Opcionais e xsd:choice

DFHWS2LS e DFHSC2LS suportam o uso de maxOccurs e minOccurs nos elementos <xsd:sequence>, <xsd:choice> e <xsd:all> somente no nível de mapeamento 2.1 e acima, em que os atributos minOccurs e maxOccurs são configurados como minOccurs="0" e maxOccurs="1".

Os assistentes geram mapeamentos que tratam estes elementos como se cada elemento filho neles fosse opcional. Ao implementar um aplicativo com estes elementos, assegure que combinações inválidas de opções não sejam geradas pelo aplicativo. Cada um dos elementos possui seu próprio campo count na estrutura de linguagens gerada, todos estes campos devem ser configurados como "0" ou todos devem ser configurados como "1". Qualquer outra combinação de valores é inválida, exceto para elementos <xsd:choice>.

Elementos <xsd:choice> indicam que somente uma das opções no elemento pode ser usada. Isto é suportado em todos os níveis de mapeamento. Os assistentes manipulam cada uma das opções em um <xsd:choice> como se estivesse em um elemento <xsd:sequence> com minOccurs="0" e maxOccurs="1". Tome cuidado ao implementar um aplicativo usando o elemento <xsd:choice> para assegurar que combinações inválidas de opções não sejam geradas pelo aplicativo. Cada um dos elementos possui seu próprio campo count na estrutura de linguagens gerada, exatamente um dos quais deve ser configurado como '1' e todos os outros devem

ser configurados como '0'. Qualquer outra combinação de valores é inválida, exceto quando o elemento <xsd:choice> é em si opcional, nesse caso ele é válido para todos os campos a serem configurados como '0'.

Suporte para Atributos XML

Os esquemas XML podem especificar atributos que são permitidos ou necessários no XML. Os utilitários do assistente CICS, DFHWS2LS e DFHSC2LS, ignoram atributos XML por padrão. Para processar atributos XML que são definidos no esquema XML, o valor do parâmetro **MAPPING-LEVEL** deve ser 1.1 ou superior.

Atributos Opcionais

Os atributos podem ser opcionais ou necessários e podem ser associados a qualquer elemento em uma mensagem SOAP ou XML para um aplicativo. Para cada atributo opcional definido no esquema, dois campos são gerados na estrutura de linguagem apropriada:

1. Uma sinalização de existência; este campo é tratado como um tipo de dados Booleano e geralmente tem 1 byte de comprimento.
2. Um valor; este campo é mapeado da mesma maneira que um elemento XML de tipo equivalente. Por exemplo, um atributo do tipo NMTOKEN é mapeado da mesma maneira que um elemento XML do tipo NMTOKEN.

Os campos de existência de atributo e de valor aparecem na estrutura de linguagem gerada antes do campo para o elemento ao qual eles estão associados. Atributos inesperados que aparecem no documento da instância são ignorados.

Por exemplo, considere a definição de atributo de esquema a seguir:

```
<xsd:attribute name="age" type="xsd:short"
use="optional" />
```

Este atributo opcional mapeia para a estrutura COBOL a seguir:

```
05 attr-age-exist PIC X DISPLAY
05 attr-age-value PIC S9999 COMP-5 SYNC
```

Processamento de Tempo de Execução de Atributos Opcionais

O processamento de tempo de execução a seguir ocorre para atributos opcionais:

- Se o atributo estiver presente, a sinalização de existência será configurada e o valor será mapeado.
- Se o atributo não estiver presente, o sinalizador de existência não será configurado.
- Se o atributo tiver um valor padrão e estiver presente, o valor será mapeado.
- Se o atributo tiver um valor padrão e não estiver presente, o valor padrão será mapeado.

Atributos opcionais que têm valores padrão são tratados como atributos necessários.

Quando o CICS transforma os dados em XML, o processamento de execução a seguir ocorre:

- Se a sinalização de existência estiver configurada, o atributo será transformado e incluído no XML.
- Se a sinalização de existência não estiver configurada, o atributo não será incluído no XML.

Atributos Necessários e o Processamento de Tempo de Execução

Para cada atributo que é necessário, somente o campo de valor é gerado na estrutura de linguagem apropriada.

Se o atributo estiver presente no XML, o valor será mapeado. Se o atributo não estiver presente, ocorrerá o seguinte processamento:

- Se o aplicativo for um provedor de serviço da web, o CICS gerará uma mensagem de falha SOAP indicando um erro na mensagem SOAP do cliente.
- Se o aplicativo for um solicitante de serviço da web, o CICS emitirá uma mensagem e retornará uma resposta de erro de conversão com um código RESP2 igual a 13 para o aplicativo.
- Se o aplicativo estiver usando o comando **TRANSFORM XMLTODATA**, o CICS emitirá uma mensagem e retornará uma resposta de solicitação inválida com um código RESP2 igual a 3 para o aplicativo.

Quando o CICS produz uma mensagem SOAP baseada no conteúdo de uma COMMAREA ou um contêiner, o atributo é transformado e incluído na mensagem. Quando um aplicativo usa o comando **TRANSFORM DATATOXML**, o CICS também transforma o atributo e o inclui no XML.

O Atributo que Permite Nil

O atributo nillable é um atributo especial que pode aparecer em um `xsd:element` em um esquema XML. Ele especifica que o atributo `xsi:nil` é válido para o elemento no XML. Se um elemento tiver o atributo `xsi:nil` especificado, isso indica que o elemento está presente, mas não tem valor e, portanto, nenhum conteúdo está associado a ele.

Se um esquema XML definiu o atributo nillable como true, ele é mapeado como um atributo necessário que utiliza um valor Booleano.

Quando o CICS recebe uma mensagem SOAP ou precisa transformar XML para um aplicativo que contém um atributo `xsi:nil`, o valor do atributo é true ou false. Se o valor for true, o aplicativo deverá ignorar os valores do elemento ou elementos aninhados no escopo do atributo `xsi:nil`.

Quando o CICS produz uma mensagem SOAP ou XML com base no conteúdo de uma COMMAREA ou um contêiner para o qual o valor para o atributo `xsi:nil` é true, ocorre o seguinte processamento:

- O atributo `xsi:nil` é gerado no XML ou mensagem SOAP.
- O valor do elemento associado é ignorado.
- Quaisquer elementos aninhados no elemento são ignorados.

Exemplo de Mensagem SOAP

Considere o esquema XML de exemplo a seguir, que poderia ser parte de um 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>

```

Aqui está um exemplo de uma mensagem SOAP parcial que está em conformidade com este esquema:

```

<root xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <num xsi:nil="true"/>
  <num>15</num>
  <num xsi:nil="true"/>
</root>

```

No COBOL, esta mensagem SOAP mapeia para estes 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)

```

Suporte para <xsd:any> e xsd:anyType

DFHWS2LS e DFHSC2LS suportam o uso de <xsd:any> e xsd:anyType no esquema XML. É possível usar o elemento do esquema XML <xsd:any> para descrever uma seção de um documento XML com conteúdo indefinido. xsd:anyType é o tipo de dados de base a partir dos quais todos os tipos de dados simples e complexos são derivados; ele não tem restrições no conteúdo dos dados.

Antes de poder usar <xsd:any> e xsd:anyType com os assistentes CICS, configure os parâmetros a seguir:

- Configure o parâmetro **MAPPING-LEVEL** como 2.1 ou superior.
- Para um aplicativo do provedor de serviços da web, configure o parâmetro **PGMINT** como CHANNEL.

Exemplo de <xsd:any>

Este exemplo usa um elemento <xsd:any> para descrever algum conteúdo XML opcional não estruturado após a tag "Surname" na tag "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>

```

Uma mensagem SOAP de exemplo que está em conformidade com este esquema XML é:

```

<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>

```

Se esta mensagem SOAP é enviada ao CICS, o CICS preenche o contêiner Customer-xml-cont com os dados XML a seguir:

```

<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>

```

O CICS também preenche o contêiner Customer-xmlns-cont com as seguintes declarações de namespace XML que estão no escopo; essas declarações são separadas por um espaço:

```

xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns="http://www.example.org/anyExample"

```

Exemplo de xsd:anyType

O xsd:anyType é o tipo de dados de base a partir do qual todos os tipos de dados simples e complexos são derivados. Ele não restringe o conteúdo de dados. Se você não especificar um tipo de dados, ele será padronizado como xsd:anyType; por exemplo, esses dois fragmentos de XML são equivalentes:

```

<xsd:element name="Name" type="xsd:anyType"/>
<xsd:element name="Name"/>

```

Estruturas de Linguagem Geradas

As estruturas de linguagem geradas para <xsd:any> ou xsd:anyType têm o formato a seguir em COBOL e um formato equivalente para as outras linguagens:

elementName-xml-cont PIC X(16)

O nome de um contêiner que contém o XML bruto. Quando o CICS processa uma mensagem SOAP recebida, ele coloca o subconjunto da mensagem SOAP que o <xsd:any> ou o xsd:anyType define neste contêiner. O aplicativo pode processar os dados XML somente nativamente. O aplicativo deve gerar o XML, preencher este contêiner e fornecer o nome do contêiner.

Este contêiner deve ser preenchido no modo de texto. Se o CICS preenche esse contêiner, ele faz isso usando a mesma variante de EBCDIC que o serviço da web está definido para usar. Caracteres que não existem na página de códigos EBCDIC de destino são substituídos por caracteres substitutos, mesmo se o contêiner é lido pelo aplicativo em UTF-8.

elementName-xmlns-cont PIC X(16)

O nome de um contêiner que contém quaisquer declarações de prefixo de namespace que estão no escopo. O conteúdo deste contêiner é semelhante àqueles do contêiner DFHWS-XMLNS, exceto que ele inclui todas as

declarações de namespace que estão no escopo e que são relevantes, em vez de somente o subconjunto da tag SOAP Envelope.

Este contêiner deve ser preenchido no modo de texto. Se o CICS preenche esse contêiner, ele faz isso usando a mesma variante de EBCDIC que o serviço da web está definido para usar. Caracteres que não existem na página de códigos EBCDIC de destino são substituídos por caracteres substitutos, mesmo se o contêiner é lido pelo aplicativo em UTF-8.

Este contêiner é usado somente ao processar mensagens SOAP enviadas ao CICS. Se o aplicativo tentar fornecer um contêiner com declarações de namespace quando uma mensagem SOAP de saída for gerada, o contêiner e seu conteúdo serão ignorados pelo CICS. O CICS requer que o XML fornecido pelo aplicativo seja totalmente autocontido com relação a declarações de namespace.

O nome do elemento XML que contém o elemento `<xsd:any>` é incluído nos nomes de variáveis que são gerados para o elemento `<xsd:any>`. No exemplo de `<xsd:any>`, o elemento `<xsd:any>` é aninhado dentro do elemento `<xsd:element name="Customer">` e os nomes de variáveis que são gerados para o elemento `<xsd:any>` são `Customer-xml-cont PIC X(16)` e `Customer-xmlns-cont PIC X(16)`.

Para um tipo `xsd:anyType`, o nome do elemento XML direto é usado; no exemplo de `xsd:anyType` acima, os nomes de variáveis são `Name-xml-cont PIC X(16)` e `Name-xmlns-cont PIC X(16)`.

Suporte para `<xsd:choice>`

Um elemento `<xsd:choice>` indica que somente uma das opções no elemento pode ser utilizada. Os assistentes CICS fornecem graus variados de suporte para elementos `<xsd:choice>` nos vários níveis de mapeamento.

Suporte para `<xsd:choice>` no nível de mapeamento 2.2 e superior

No nível de mapeamento 2.2 e superior, DFHWS2LS e DFHSC2LS fornecem suporte aprimorado para elementos `<xsd:choice>`. Os assistentes geram um novo contêiner que armazena o valor associado ao elemento `<xsd:choice>`. Os assistentes geram estruturas de linguagem contendo o nome de um novo contêiner e um campo extra:

fieldname -enum

O campo discriminador para indicar qual das opções o elemento `<xsd:choice>` usará.

fieldname -cont

O nome do contêiner que armazena a opção a ser usada. Uma estrutura de linguagem adicional é gerada para mapear o valor da opção.

O fragmento de esquema XML a seguir inclui um 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>
```

Se este fragmento de esquema XML for processado no nível de mapeamento 2.2 ou superior, o assistente gerará as estruturas de linguagem COBOL a seguir:

```
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.
```

Limitações para <xsd:choice> no nível de mapeamento 2.2 e superior

DFHSC2LS e DFHWS2LS não suportam elementos <xsd:choice> aninhados; por exemplo, o XML a seguir não é suportado:

```
<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 e DFHWS2LS não suportam elementos <xsd:choice> recorrentes; por exemplo, o XML a seguir não é suportado:

```
<xsd:choice maxOccurs="2">  
<xsd:element name="name1" type="string"/>  
</xsd:choice>
```

DFHSC2LS e DFHWS2LS suportam, no máximo, 255 opções em um elemento <xsd:choice>.

Suporte para <xsd:choice> no nível de mapeamento 2.1 e abaixo

No nível de mapeamento 2.1 e abaixo, o DFHWS2LS fornece suporte limitado para elementos <xsd:choice>. O DFHWS2LS trata cada uma das opções em um elemento <xsd:choice> como se fossem um elemento <xsd:sequence> que pode ocorrer no máximo uma vez.

Apenas uma das opções em um elemento <xsd:choice> pode ser usada, portanto, tenha cuidado ao implementar um aplicativo usando o elemento <xsd:choice> gerado somente com combinações válidas de opções. Cada um dos elementos possui seu próprio campo count na estrutura de linguagens gerada, exatamente um dos quais deve ser configurado como 1 e todos os outros devem ser configurados como 0. Qualquer outra combinação de valores é incorreta, exceto quando o <xsd:choice> é em si opcional, nesse caso é válido que todos os campos sejam configurados como 0.

Suporte para <xsd:sequence>

O uso de elementos <xsd:sequence> com os assistentes do CICS é suportado com inúmeras limitações.

O uso dos atributos minOccurs e maxOccurs não é suportado para o elemento <xsd:sequence>. As exceções a essa regra são quando minOccurs="0" e maxOccurs="1" ou minOccurs="1" e maxOccurs="1".

Os assistentes do CICS não permitem que dois elementos com o mesmo nome sejam usados no mesmo elemento <xsd:sequence>. Por exemplo, a sequência {a, b, c, a} é rejeitada pelos Assistentes do CICS. Para evitar esta limitação, substitua a sequência por {a maxOccurs="2", b, c}.

Os Assistentes do CICS permitem somente um elemento <xsd:any>, ou um elemento que é tratado como um <xsd:any>, no mesmo elemento <xsd:sequence>. Elementos que são tratados como elementos <xsd:any> incluem elementos abstratos que não possuem um grupo de substituição definido.

Ao analisar o XML, o CICS trata elementos <xsd:sequence> como elementos <xsd:all>; isso significa que o CICS não verifica se a ordem dos itens na sequência está correta. Por exemplo, se o esquema definir uma sequência igual a {a, b, c}, o CICS tolerará XML que contém a, b e c em qualquer ordem. No entanto, quando o CICS gera XML, a ordem dos itens em um <xsd:sequence> é sempre preservada.

O CICS não detecta automaticamente dados XML ausentes. Por exemplo, se um elemento no <xsd:sequence> for definido como obrigatório, (minOccurs="1" maxOccurs="1"), mas não ocorrer no documento XML, o CICS relatará esse problema somente se a validação de tempo de execução estiver ativada

Suporte para Grupos de Substituição

É possível usar um grupo de substituição para definir um grupo de elementos XML que são intercambiáveis. Os assistentes do CICS fornecem suporte para grupos de substituição no nível de mapeamento 2.2 e superior.

No nível de mapeamento 2.2 e superior, DFHSC2LS e DFHWS2LS suportam grupos de substituição usando mapeamentos semelhantes àqueles usados para elementos <xsd:choice>. O assistente gera um campo de enumeração e um novo nome de contêiner na estrutura de linguagem.

O fragmento de esquema XML a seguir inclui uma matriz de dois elementos subGroupParent, cada um dos quais pode ser substituído por replacementOption1 ou 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" />
```

Processar este fragmento XML com o assistente gera as estruturas de linguagem COBOL a seguir:

```
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 obter mais informações sobre grupos de substituição, consulte a *especificação Esquema XML W3C Parte 1: Estruturas Segunda Edição*: http://www.w3.org/TR/xmlschema-1/#Elements_Equivalence_Class

Suporte para Elementos Abstratos e Tipos de Dados Abstratos

Os assistentes do CICS fornecem suporte para elementos abstratos e tipos de dados abstratos no nível de mapeamento 2.2 e superior. Os assistentes do CICS mapeiam elementos abstratos e tipos de dados abstratos de uma maneira semelhante a grupos de substituição.

Suporte para elementos abstratos no nível de mapeamento 2.2 e superior

No nível de mapeamento 2.2 e superior, DFHSC2LS e DFHWS2LS tratam elementos abstratos quase da mesma maneira que grupos de substituição, exceto que o elemento abstrato não é um membro válido do grupo. Se não houver elementos substituíveis, o elemento abstrato será tratado como um elemento `<xsd:any>` e usará os mesmos mapeamentos que um elemento `<xsd:any>` no nível de mapeamento 2.1.

O fragmento de esquema XML a seguir especifica duas opções que podem ser usadas no lugar do elemento abstrato. O elemento abstrato em si não é uma opção 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" />
```

Processar este fragmento XML com o assistente gera as estruturas de linguagem COBOL a seguir:

```
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 obter mais informações sobre elementos abstratos, consulte a *especificação Esquema XML W3C Parte 0: Manual Segunda Edição*: <http://www.w3.org/TR/xmlschema-0/#SubsGroups>

Suporte para tipos de dados abstratos no nível de mapeamento 2.2 e superior

No nível de mapeamento 2.2 e superior, DFHSC2LS e DFHWS2LS tratam tipos de dados abstratos como grupos de substituição. O assistente gera um campo de enumeração e um novo nome de contêiner na estrutura de linguagem.

O fragmento de esquema XML a seguir especifica duas alternativas que podem ser usadas no lugar do tipo abstrato:

```
<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>
```

Processar este fragmento XML com o assistente gera as estruturas de linguagem COBOL a seguir:

```
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).
```

As estruturas de linguagem são geradas em copybooks separados. A estrutura de linguagem gerada para option1 é gerada em um copybook:

```
03 option1 PIC X(5).
```

A estrutura de linguagem para option2 é gerada em um copybook diferente:

```
03 option2 PIC X(10).
```

Para obter mais informações sobre tipos de dados abstratos, consulte a *especificação W3C XML Schema Parte 0: Primer Second Edition* : <http://www.w3.org/TR/xmlschema-0/#SubsGroups>

Como Manipular Conteúdo Variavelmente Repetido no COBOL

No COBOL, não é possível processar conteúdo variavelmente repetido usando aritmética do ponteiro para endereçar cada instância dos dados. Outras linguagens de programação não possuem esta limitação. Este exemplo mostra como manipular o conteúdo de repetição variável no COBOL para um aplicativo de serviço da web.

Esta técnica também se aplica na transformação do XML em dados do aplicativo usando os comandos da API **TRANSFORM**. O documento WSDL de exemplo a seguir representa um serviço da web com dados do aplicativo que consistem em uma sequência de 8 caracteres que ocorre um número variável de vezes:

```
<?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>

```

O processamento deste documento WSDL por meio do DFHWS2LS gera as estruturas de linguagem COBOL a seguir:

```
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).
```

Observe que o campo component de 8 caracteres é definido em uma estrutura separada chamada DFHWS-component. A estrutura de dados principal é chamada de applicationData e contém dois campos, component-num e component-cont. O campo component-num indica quantas instâncias dos dados component estão presentes e o campo component-cont indica o nome de um contêiner que contém a lista concatenada de campos component.

O código COBOL a seguir demonstra uma maneira de processar a lista de dados variavelmente recorrentes. Ele faz uso de uma matriz de seção de ligação para endereçar instâncias subsequentes dos dados, cada uma das quais é exibida usando a instrução DISPLAY:

```
IDENTIFICATION DIVISION.
PROGRAM-ID. EXVARY.
```

```
ENVIRONMENT DIVISION.
DATA DIVISION.
SEÇÃO DE ARMAZENAMENTO DE FUNCIONAMENTO.
```

```
* working storage variables
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.
```

```
* a large linkage section array
01 BIG-ARRAY PIC X(659999).
```

```
* application data structures produced by DFHWS2LS
* this is normally referenced with a COPY statement
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.
```

```
* Get the DFHWS-DATA container
```

```

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

* Get the recurring component data
EXEC CICS GET CONTAINER(component-cont)
SET(COMPONENT-PTR)
FLENGTH(COMPONENT-DATA-LENGTH)
END-EXEC

* Point the component structure at the first instance of the data
SET ADDRESS OF DFHWS-component TO COMPONENT-PTR

* Store the length of a single component
MOVE LENGTH OF DFHWS-component TO COMPONENT-LENGTH

* process each instance of component data in turn
PERFORM WITH TEST AFTER
UNTIL COMPONENT-COUNT = component-num

* display the current instance of the data
DISPLAY 'component value is: ' component

* address the next instance of the component data
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

* end the loop
END-PERFORM.

* Point the component structure back at the first instance of
* of the data, for any further processing we may want to perform
SET ADDRESS OF DFHWS-component TO COMPONENT-PTR

* return to CICS.

EXEC CICS
RETURN
END-EXEC

GOBACK.

```

O código acima fornece uma solução genérica para manipular conteúdo de repetição variável. A matriz, BIG-ARRAY, move para o início de cada componente por vez e não permanece fixa no início dos dados. A estrutura de dados do componente é, então, movida para o ponto no primeiro byte do próximo componente. COMPONENT-PTR pode ser usado para recuperar a posição inicial dos dados do componente, se necessário.

Aqui está uma mensagem SOAP de exemplo que está em conformidade com o 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>
```

Aqui está a saída produzida pelo programa COBOL quando ele processa a mensagem SOAP:

```
CPIH 20080115103151 component value is: VALUE1
CPIH 20080115103151 component value is: VALUE2
CPIH 20080115103151 component value is: VALUE3
```

Suporte para Valores de Comprimento Variável e Espaço em Branco

É possível customizar a maneira na qual os valores de comprimento variável e espaços em branco são manipulados usando as configurações nos assistentes CICS e incluindo máscaras diretamente no esquema XML.

Geralmente, o assistente CICS XML e o assistente de serviço da web do CICS mapeiam sequências de dados para matrizes de caracteres de comprimento fixo; essas matrizes requerem preenchimento com espaços ou nulos. O mapeamento de valores de comprimento variável para matrizes de dados de comprimento fixo pode ser ineficiente e desperdiçar armazenamento. Se o comprimento de seus dados é variável, é recomendável customizar a maneira como esses mapeamentos são manipulados.

Se você estiver convertendo a partir de uma estrutura de linguagem para um esquema XML ou documento WSDL, é recomendável especificar as máscaras `whiteSpace` e `maxLength` em seu esquema XML e configurar o parâmetro **CHAR-VARYING-LIMIT** nos assistentes.

Se você estiver convertendo a partir de um esquema XML ou documento WSDL para uma estrutura de linguagem, é recomendável configurar um valor apropriado para o parâmetro **CHAR-VARYING** nos assistentes.

Nota: Caracteres nulos ('x00') não são válidos em documentos XML. Quaisquer caracteres nulos de dados do aplicativo analisados pelo CICS são vistos como o fim de uma sequência e o valor é truncado. Quando o CICS gera dados do aplicativo, ele faz isso de acordo com o valor do parâmetro **CHAR-VARYING**. Por exemplo, se a opção **CHAR-VARYING=NULL** for especificada, as sequências de comprimento variável geradas pelo CICS serão finalizadas com um caractere nulo.

Mapeando valores de comprimento variável a partir do XML para estruturas de linguagem

Use máscaras no esquema XML ou especifique determinados parâmetros nos assistentes CICS para customizar a maneira na qual os mapeamentos entre o esquema XML ou o documento WSDL e a estrutura de linguagem são manipulados.

Tipos de dados XML podem ser restringidos usando máscaras. Use as máscaras de comprimento (`length`, `maxLength` e `minLength`) e a máscara `whiteSpace` para customizar a maneira como os dados de comprimento variável em seu XML são manipulados.

length

Usado para especificar que os dados são de comprimento fixo.

maxLength

Usado para especificar o comprimento máximo para o tipo de dados. Se esse valor não for configurado para um tipo de dados baseado em sequência, o comprimento máximo será ilimitado.

minLength

Usado para especificar o comprimento mínimo para o tipo de dados. Se esse valor não for configurado para um tipo de dados baseado em sequência, o comprimento mínimo será 0.

whiteSpace

Usado para especificar como o espaço em branco ao redor de um valor de dados é manipulado. O espaço em branco inclui espaços, tabulações e novas linhas. A máscara whiteSpace pode ser configurada como preserve, replace ou collapse:

- Um valor preserve mantém qualquer espaço em branco no valor dos dados.
- Um valor replace significa que quaisquer tabulações ou novas linhas serão substituídas pelo número apropriado de espaços.
- Um valor collapse significa que espaços em branco iniciais, finais e integrados são removidos e que todas as tabulações, novas linhas e espaços consecutivos são substituídos por caracteres de espaço único.

Se a máscara whiteSpace não for configurada, o espaço em branco será preservado.

Para obter mais informações sobre as máscaras de esquema XML, consulte o esquema de recomendação do W3C *Esquema XML Parte 2: Tipos de Dados Segunda Edição* <http://www.w3.org/TR/xmlschema-2/#facets>

Os parâmetros a seguir nos assistentes CICS, DFHSC2LS e DFHWS2LS, podem ser usados para alterar a maneira como os dados de comprimento variável são mapeados a partir do esquema XML para a estrutura de linguagem. Esses parâmetros estão disponíveis no nível de mapeamento 1.2 ou superior.

DEFAULT-CHAR-MAXLENGTH

Especifica o comprimento da matriz padrão de dados de caractere em caracteres para mapeamentos em que nenhum comprimento está implícito no esquema XML ou documento WSDL. O valor desse parâmetro pode ser um número inteiro positivo no intervalo de 1 a 2147483647.

No entanto, é recomendável especificar o comprimento máximo de caracteres que você deseja que DFHSC2LS ou DFHWS2LS use diretamente em seu esquema XML ou documento WSDL com a máscara maxLength. Especificar o comprimento máximo diretamente no esquema XML ou documento WSDL evita problemas associados a ter um padrão global aplicado a todos os tipos de dados baseados em sequência.

CHAR-VARYING-LIMIT

Especifica o tamanho máximo dos dados de caractere de comprimento variável que são mapeados para a estrutura de linguagem. Se os dados de caractere forem maiores que o valor especificado neste parâmetro, eles serão mapeados para um contêiner e o nome do contêiner será usado na estrutura de linguagem gerada. O valor pode variar de 0 até o padrão de 32767 bytes.

CHAR-VARYING

Especifica como os dados de caracteres de comprimento variável são

mapeados. Se você não especificar esse parâmetro, o mapeamento padrão dependerá da linguagem especificada. Você pode selecionar estas opções:

- **CHAR-VARYING=NO** especifica que os dados de caracteres de comprimento variável são mapeados como sequências de comprimento fixo.
- **CHAR-VARYING=NULL** especifica que dados de caracteres de comprimento variável são mapeados para sequências com terminação nula.
- **CHAR-VARYING=YES** especifica que dados de caracteres de comprimento variável são mapeados para um tipo de dados CHAR VARYING em PL/I. Nas linguagens COBOL, C e C++, os dados de caracteres de comprimento variável são mapeados para uma representação equivalente que consiste em dois elementos relacionados: o comprimento dos dados e os dados.

A configuração de **CHAR-VARYING=YES** geralmente resulta no melhor desempenho.

Mapeando valores de comprimento variável a partir de estruturas de linguagem para XML

É possível customizar a maneira na qual os mapeamentos entre a estrutura de linguagem e o esquema XML, ou documento WSDL, são manipulados. Configure o parâmetro **CHAR-VARYING** em DFHLS2SC ou DFHLS2WS para **COLLAPSE** ou **NULL** para mudar a maneira como as matrizes de caracteres são geradas.

A configuração da opção **CHAR-VARYING=NULL** informa ao CICS para incluir um caractere nulo no final de cada matriz de caracteres ao gerar XML.

A configuração da opção **CHAR-VARYING=COLLAPSE** informa ao CICS para remover automaticamente quaisquer espaços à direita do final das matrizes de caracteres ao gerar o XML. Essa opção está disponível somente no nível de mapeamento 2.1 ou superior e **CHAR-VARYING=COLLAPSE** é o valor padrão no nível de mapeamento 2.1 ou superior para todas as linguagens diferentes de C e C++. Quando o XML é analisado, todos os espaços em branco iniciais, finais e integrados são removidos.

Para obter mais informações, consulte Suporte para espaço em branco e valores de comprimento variável nos serviços da web do CICS (nota técnica) .

Suporte para UTF-16 em dados do aplicativo

Os serviços da web CICS suportam a conversão de dados do aplicativo codificados em UTF-16 em XML ou JSON e também XML ou JSON em dados do aplicativo codificados em UTF-16. Use o UTF-16 quando você precisa armazenar e processar dados em diversos idiomas.

Os serviços da web CICS SOAP e JSON suportam a conversão de dados do aplicativo codificados em UTF-16 em XML ou JSON e também XML ou JSON em dados do aplicativo codificados em UTF-16. O Unicode é um esquema de codificação de largura variável que permite que os sistemas manipulem dados de forma eficiente.

UTF-16 é uma codificação de largura variável para Unicode, em que cada caractere é representado por 2 ou 4 bytes. Os serviços da web do CICS suportam CCSID 1200 para dados do aplicativo, que são UTF-16 BE (big endian) com a Área de Uso Privado da IBM. Esse comportamento é consistente com o suporte UTF-16 em todas as linguagens suportadas.

UTF-16 é suportado no nível de mapeamento 4.0 e superior. É possível customizar como os dados do aplicativo são convertidos usando configurações de mapeamento nos assistentes. Para obter mais informações sobre níveis de mapeamento XML, consulte Níveis de mapeamento para os assistentes CICS. Para obter mais informações sobre níveis de mapeamento JSON, consulte Níveis de mapeamento para os assistentes CICS JSON.

Nota: UTF-16 requer mais tempo de processamento e tem menos eficiência de armazenamento do que as codificações EBCDIC. Além disso, a combinação de tipos de codificação incorre em processamento de tempo de execução extra.

Mapeando UTF-16 do esquema XML ou JSON para estruturas de linguagem

O suporte para UTF-16 depende de como você cria o serviço da web. O mapeamento de esquema XML ou JSON para estruturas de linguagem, também conhecido como mapeamento de cima para baixo, tem as características a seguir. Se UTF-16 está ativado, todos os campos de texto são mapeados para campos UTF-16, enquanto que tipos de dados de exibição numéricos em COBOL são mapeados como EBCDIC. Para usar UTF-16, configure o parâmetro CCSID de DFHJS2LS, DFHSC2LS ou DFHWS2LS para 1200.

Por exemplo, se o fragmento de esquema XML a seguir estiver presente no WSDL:

```
<xsd:element name="myString" nillable="false">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:maxLength value="20"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

O assistente DFHWS2LS gerará o campo a seguir em uma estrutura de linguagem COBOL:

```
myString PIC N(
20
) USAGE NATIONAL
```

O parâmetro CHAR-MULTIPLIER dos assistentes de serviços da web pode ser usado para especificar o comprimento de um campo que os assistentes geram.

CHAR-MULTIPLIER

Quando você usa UTF-16, os únicos valores válidos para o parâmetro **CHAR-MULTIPLIER** são 2 ou 4, em que 2 é o valor padrão.

CHAR-MULTIPLIER = 2, em que o esquema descreve uma sequência de $\text{maxlength} \times$, gera PIC N(x). Configurar **CHAR-MULTIPLIER** = 2 não impede o uso de pares substitutos em uma sequência UTF-16, mas afeta o número de caracteres que se ajustam no campo.

CHAR-MULTIPLIER = 4 gera PIC N($2x$). Se **CHAR-MULTIPLIER** = 4, o valor no tempo de execução é preenchido se a sequência inclui caracteres que podem ser expressos em uma única unidade de codificação.

Mapeando UTF-16 a partir de estruturas de linguagem para esquema XML ou JSON

O mapeamento de uma estrutura de linguagem para o esquema XML ou JSON, também conhecido como mapeamento bottom-up, é gerenciado de forma diferente do mapeamento de cima para baixo. Se uma sequência UTF-16 for declarada na estrutura de linguagem, os dados serão interpretados pelo CICS como codificados em UTF-16, caso contrário, assume-se que os dados estejam em uma codificação EBCDIC. O parâmetro CCSID para DFHLS2JS, DFHLS2SC ou DFHLS2WS indica a codificação de qualquer texto EBCDIC nos dados do aplicativo; ele não deve ser configurado para indicar UTF-16.

Os tipos de dados que são interpretadas como caracteres UTF-16 são os seguintes: PIC N (*n*) em COBOL, WIDECHAR(*n*) em PL/I e char16_t[*n*] em C e C++.

O parâmetro CHAR-USAGE dos assistentes de serviços da web pode ser usado para especificar os tipos de dados.

CHAR-USAGE

Em COBOL, o tipo de dado nacional, PIC N, pode ser usado para dados UTF-16 ou DBCS. Essa configuração é controlada pela opção do compilador NSYMBOL. Deve-se configurar o parâmetro **CHAR-USAGE** no assistente com o mesmo valor que a opção do compilador NSYMBOL para assegurar que os dados sejam manipulados apropriadamente. Isso geralmente é configurado como CHAR-USAGE=NATIONAL quando você usa UTF-16.

Se você deseja combinar tipos de dados nacionais que contêm dados UTF-16 e DBCS no mesmo copybook, é possível usar os qualificadores USAGE NATIONAL ou USAGE DISPLAY-1 em campos individuais.

Nota: DFHLS2WS, DFHLS2SC e DFHLS2JS não suportam a cláusula COBOL GROUP USAGE NATIONAL.

Consultando XML a partir de um aplicativo

É possível gravar um programa aplicativo para consultar um fragmento de XML antes de transformá-lo em dados do aplicativo.

Sobre Esta Tarefa

Se seu aplicativo processará muitos tipos diferentes de XML, talvez você deseje consultar o XML para determinar qual recurso XMLTRANSFORM usar para transformá-lo em dados do aplicativo. Este comando também pode ser útil se seu XML contém elementos <xsd:any>.

Procedimento

1. Em seu programa de aplicativo, use o comando **TRANSFORM XMLTODATA** API para consultar o XML:

```
EXEC CICS TRANSFORM XMLTODATA  
CHANNEL('MyChannelName')  
XMLCONTAINER('SourceContainerName')  
ELEMNAME(elementName) ELEMNAMELEN(elementNameLength)
```

Deve-se especificar o nome do canal e o nome do contêiner que contém o XML. Você não precisa especificar um recurso XMLTRANSFORM para consultar o XML. O exemplo acima consulta o nome do primeiro elemento XML e o

comprimento do elemento XML. Também é possível consultar o tipo do primeiro elemento XML, o comprimento do tipo e o namespace do tipo.

2. Opcional: Se o aplicativo requerer o namespace do elemento XML, forneça uma área de dados na qual o CICS possa gravar o valor ELEMNS.
3. Opcional: Após consultar o XML, é possível gravar a lógica de aplicativo para determinar qual recurso XMLTRANSFORM usar para transformar o XML em dados do aplicativo.
4. Instale o programa de aplicativo no CICS.

Resultados

O CICS lê o contêiner especificado e retorna as informações sobre o elemento XML ao programa de aplicativo.

Manipulando XML por tipo de dados

Se o esquema XML contém tipos de dados globais, um ou mais dos quais são referenciados no XML, é possível gerar metadados para suportar esses tipos de dados globais e, em seguida, analisar ou transformar o XML em um programa de aplicativo.

Antes de Iniciar

Para gerar os metadados corretos, deve-se executar o DFHSC2LS especificando o parâmetro TYPES=ALL.

Sobre Esta Tarefa

Procedimento

1. Para transformar dados do aplicativo em XML por tipo de dados, use o comando **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 é o nome com 32 caracteres do recurso XMLTRANSFORM que especifica a ligação XML e o esquema; *MyChannelName* é o nome com 16 caracteres do canal que possui os contêineres de entrada e saída; *SourceContainerName* é o nome com 16 caracteres do contêiner de entrada que contém os dados do aplicativo e *TargetContainerName* é o nome com 16 caracteres do contêiner de saída que o CICS preenche com XML. Você também deve especificar o nome do elemento XML, o namespace, o tipo de dados e o namespace do tipo de dados para a transformação.

2. Para analisar o XML por tipo de dados, use o comando **TRANSFORM XMLTODATA**. As opções que você especifica no comando dependem de se o XML possui um atributo `xsi:type`.

- Se o XML usa o atributo `xsi:type`, especifique o comando a seguir em seu programa de aplicativo:

```

EXEC CICS TRANSFORM XMLTODATA
XMLTRANSFORM('
MyXmlTransformName
')
CHANNEL('
MyChannelName
')
XMLCONTAINER('
SourceContainerName
')
DATCONTAINER('
TargetContainerName
')
ELEMNAME(elementName) ELEMNAMELEN(elementNameLength)
TYPENAME(typeName) TYPENAMELEN(typeNameLength)

```

Se o aplicativo também requer o namespace do tipo de dados, inclua a opção `TYPENS` no comando de API. O CICS retorna o tipo de dados do atributo `xsi:type` no `TYPENAME`.

- Se o XML não usa o atributo `xsi:type`, o programa de aplicativo pode especificar o nome local e o namespace do tipo de dados global. Use o comando a seguir em seu programa de aplicativo:

```

EXEC CICS TRANSFORM XMLTODATA
XMLTRANSFORM('
MyXmlTransformName
')
CHANNEL('
MyChannelName
')
XMLCONTAINER('
SourceContainerName
')
DATCONTAINER('
TargetContainerName
')
TYPENAME(
typeName
) TYPENAMELEN(
typeNameLength
)
TYPENS(

```

```

TypeNamespace
) TYPENSLEN(
typeNamespaceLen
)

```

O programa de aplicativo deve especificar o nome do tipo e o namespace, mas não o nome e o namespace. Esse comando indica que o aplicativo informa ao CICS qual tipo de dados usar. Quando o CICS está gerando XML, ele sempre inclui o atributo `xsi:type`.

3. Instale o programa de aplicativo no CICS.

O que Fazer Depois

Teste se o programa de aplicativo gera e analisa o XML conforme o esperado.

Manipulando tipos de dados <xsd:any>

Se você estiver trabalhando com um esquema XML que contém um ou mais tipos de dados <xsd:any>, os assistentes XML podem mapear o tipo de dados para um par de contêineres CICS. É possível gravar um programa de aplicativo para analisar o XML nos contêineres.

Antes de Iniciar

Deve-se mapear o esquema XML usando DFHSC2LS ou DFHWS2LS usando um nível de mapeamento de 2.1 ou superior.

Sobre Esta Tarefa

Quando o CICS transforma os dados em XML, ele coloca o XML associado ao tipo de dados <xsd:any> no primeiro contêiner e as declarações de prefixo de namespace que estão no escopo no segundo contêiner.

Procedimento

1. Para analisar os dados XML, use o comando **TRANSFORM XMLTODATA** em seu programa de aplicativo:

```

EXEC CICS TRANSFORM XMLTODATA
XMLTRANSFORM('
MyXmlTransformName
')
CHANNEL('
MyChannelName
')
XMLCONTAINER('
SourceContainerName
')
DATCONTAINER('
TargetContainerName
')
NSCONTAINER('
NamespacesContainerName
')
ELEMNAME(elementName) ELEMNAMELEN(elementNameLength)

```

MyXmlTransformName é o nome com 32 caracteres do recurso XMLTRANSFORM que especifica a ligação XML e o esquema; *MyChannelName* é o nome com 16 caracteres do canal que possui os contêineres de entrada e saída; *SourceContainerName* é o nome com 16 caracteres do contêiner de entrada que contém o XML e *TargetContainerName* é o nome com 16 caracteres do contêiner de saída que o CICS preenche com os dados do aplicativo;

NamespacesContainerName é o nome com 16 caracteres do contêiner que o CICS preenche com as declarações de prefixo de namespace. Forneça valores iniciais para as opções ELEMNAME e ELEMNAMELEN. O CICS retorna o elemento XML e seu comprimento nas opções ELEMNAME e ELEMNAMELEN.

2. Instale o programa de aplicativo no CICS.

O que Fazer Depois

Teste se o programa de aplicativo analisa o XML corretamente.

Validando transformações de XML

Quando você usa o assistente XML do CICS para mapear dados do aplicativo em XML, é possível especificar que as transformações que ocorrem no tempo de execução sejam validadas para assegurar que estejam em conformidade com o esquema que está contido na ligação XML. É possível executar a validação na transformação de XML em dados binários ou de dados binários em XML.

Antes de Iniciar

Durante o desenvolvimento e teste de seu aplicativo CICS, a validação completa auxilia na detecção de problemas no XML. No entanto, a validação completa do XML carrega uma sobrecarga substancial e é desaconselhável validar XML em um aplicativo de produção totalmente testado.

O CICS usa um programa Java para validar o XML com relação ao esquema. Portanto, deve-se ter suporte Java ativado em sua região CICS para executar a validação.

Procedimento

1. Configure um servidor JVM na região CICS. A classe do validador XML pode ser executada em uma estrutura OSGi ou Axis2, mas não em um perfil Liberty. O CICS fornece amostras para configurar rapidamente um servidor JVM que usa uma estrutura OSGi.
 - a. Instale o servidor JVM de amostra DFHJVMS no grupo DFH\$OSGI ou crie seu próprio servidor JVM. Para obter mais informações, consulte Configurando um Servidor JVM.
 - b. Se você criou seu próprio servidor JVM, modifique a definição do programa DFHPIVAL no grupo DFHPIVAL para referenciar o nome do recurso JVMSERVER. A definição DFHPIVAL não é bloqueada e pode ser editada. Por padrão, a definição faz referência ao DFHJVMS.
2. Assegure que a ligação XML e o esquema estejam no mesmo local no z/OS UNIX. O recurso XMLTRANSFORM define esses arquivos no CICS. É possível usar o comando **INQUIRE XMLTRANSFORM** para verificar o local de cada arquivo.
3. Ative a validação para o aplicativo. No CICS Explorer, abra o recurso XMLTRANSFORM e edite o campo Status de Validação na lista de atributos. Como alternativa, é possível usar CEMT ou SPI.

Resultados

Verifique o log do sistema para descobrir se a transformação XML é válida. A mensagem DFHML0508 indica que o XML foi validado com sucesso e a mensagem DFHML0507 indica que a validação falhou.

O que Fazer Depois

Quando a validação XML não for mais necessária para o aplicativo, atualize o recurso XMLTRANSFORM para desativá-la.

Gerando mapeamentos a partir de estruturas de linguagem

Para criar XML a partir de dados do aplicativo ou vice-versa, crie os mapeamentos para descrever como o CICS transformará os dados e o XML no tempo de execução. É possível iniciar a partir de qualquer registro de dados do aplicativo; por exemplo, é possível iniciar com uma COMMAREA, um arquivo VSAM, uma fila de armazenamento temporário ou um registro do DB2.

Antes de Iniciar

Antes de criar os mapeamentos, você deve se certificar de que estas condições tenham sido concluídas:

- Você deve ter uma estrutura de linguagem que descreva o registro de aplicativo em um conjunto de dados particionado. A estrutura de linguagem pode ser gravada em qualquer uma das linguagens de alto nível suportadas pelo assistente XML do CICS: COBOL, PL/I, C e C++. Se esse mapeamento for para uso com um feed Atom, e você estiver usando qualquer um dos campos em seu registro de aplicativo para fornecer metadados para as entradas Atom (tal como o nome de um autor), certifique-se de que esses campos não sejam aninhados em sua estrutura de linguagem. É possível ter estruturas de campos aninhados dentro de um campo que fornece o conteúdo para uma entrada Atom.
- Você deve configurar o ID do usuário sob o qual DFHLS2SC é executado para usar z/OS UNIX.
- O ID do usuário deve ter permissão de leitura para acessar a estrutura de linguagem e permissão de gravação para colocar a saída nos diretórios apropriados no z/OS UNIX.
- Você deve alocar armazenamento suficiente para o ID do usuário para que o ID execute Java. É possível usar qualquer versão suportada do Java. Por padrão, o DFHLS2SC usa a versão Java especificada no parâmetro **JAVADIR**.

Sobre Esta Tarefa

Use o assistente CICS XML para criar os mapeamentos de dados para o registro de aplicativo. Para cada linguagem de alto nível suportada pelo assistente CICS XML, alguns tipos de dados são restritos ou não suportados. O assistente CICS XML emite mensagens de erro sobre quaisquer itens suportados que ele identifica em sua estrutura de linguagem. As informações de referência para o assistente CICS XML lista as restrições que se aplicam a cada linguagem de alto nível.

Procedimento

1. Execute a tarefa em lote DFHLS2SC. DFHLS2SC possui parâmetros opcionais que você seleciona para atender aos seus requisitos, tal como selecionar uma página de códigos ou um espaço de nomes específico. Use os parâmetros a seguir como um mínimo:
 - a. Especifique o idioma de alto nível de sua estrutura de linguagem no parâmetro **LANG**.
 - b. Se você estiver implementando os mapeamentos de dados em um pacote configurável, especifique o nome de um recurso de pacote configurável no parâmetro **BUNDLE**. Se você estiver criando uma ligação XML para um feed Atom, não especifique esse parâmetro.

- c. Especifique o nível de mapeamento no parâmetro **MAPPING-LEVEL**. Se você estiver criando uma ligação XML para um feed Atom, deverá usar um nível de mapeamento igual a 3.0 ou superior. Para outras situações, embora você possa usar qualquer nível de mapeamento, para obter as opções de mapeamento mais avançadas, use o nível de mapeamento mais recente.
- d. Opcional: Se você estiver criando uma ligação XML para um feed Atom, e o registro de dados do aplicativo contiver os registros de data e hora no formato CICS ABSTIME, especifique o parâmetro opcional **DATETIME=PACKED15** para mapear esses campos como registros de data e hora.
- e. Especifique o local e a página de códigos das estruturas de linguagem que descrevem o registro de aplicativo nos parâmetros **PDSMEM** e **PDSCP**.
- f. Especifique o nome e o local do arquivo de esquema no parâmetro **SCHEMA**. A extensão do arquivo é .xsd. Se você estiver criando um pacote configurável, não especifique um local. O DFHLS2SC criará o esquema XML, mas não a estrutura de diretório, se o arquivo ainda não existir.
- g. Especifique o nome e o local da ligação XML no parâmetro **XSDBIND**. A extensão do arquivo é .xsdbind. Se você estiver criando um pacote configurável, não especifique um local. O DFHLS2SC criará a ligação XML, mas não a estrutura de diretório, se o arquivo ainda não existir.

Dica: Coloque a ligação XML e o esquema na mesma estrutura de diretório para ativar a validação. A validação pode ser útil quando você está testando seu aplicativo em um ambiente de desenvolvimento ou de teste. Se você está criando um pacote configurável, o CICS coloca os arquivos no mesmo diretório para você.

Se você especificar o parâmetro **BUNDLE**, a tarefa em lote criará uma estrutura de diretório do pacote configurável no z/OS UNIX. O diretório do pacote configurável possui um subdiretório META-INF que contém o manifest do pacote configurável. A tarefa em lote também cria um esquema XML e uma ligação XML no diretório do pacote configurável, usando os nomes de arquivo que você especificou para os parâmetros **SCHEMA** e **XSDBIND**. Se você não especificar o parâmetro **BUNDLE**, a tarefa em lote criará o esquema XML e a ligação XML somente no local especificado.

2. Instale o recurso BUNDLE ou um recurso ATOMSERVICE que especifica essa ligação XML. Os recursos BUNDLE e ATOMSERVICE criam dinamicamente um recurso XMLTRANSFORM, que define o local do esquema XML e do arquivo de ligação.

Resultados

Quando você gera mapeamentos a partir de estruturas de linguagem, somente uma transformação XML é possível.

Exemplo

O exemplo a seguir mostra DFHLS2SC com o conjunto mínimo de parâmetros 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
/*

```

O que Fazer Depois

Grave um programa de aplicativo para transformar os dados do aplicativo em XML e vice-versa. É possível usar os mesmos mapeamentos para ambas as transformações. Se você tiver criado uma ligação XML para um feed Atom, continue com as etapas para configurar seu feed Atom.

Gerando mapeamentos a partir de um esquema XML

Para criar dados do aplicativo a partir de XML ou vice-versa que está em conformidade com um esquema XML existente, você cria os mapeamentos para descrever como o CICS transformará os dados no tempo de execução. É possível iniciar a partir de um esquema XML ou um documento WSDL.

Antes de Iniciar

Deve-se ter um esquema XML ou documento WSDL válido. Antes de criar os mapeamentos, você deve se certificar de que estas condições prévias tenham sido concluídas:

- Deve-se ter um esquema XML ou documento WSDL válido.
- Deve-se configurar o ID do usuário sob o qual o DFHSC2LS é executado para usar UNIX System Services.
- O ID do usuário deve ter permissão de leitura para acessar o esquema XML ou documento WSDL e permissão de gravação para colocar a saída nos diretórios apropriados no z/OS UNIX.
- O ID do usuário deve ter uma alocação de armazenamento suficientemente grande para executar Java. É possível usar qualquer versão suportada do Java. Por padrão, o DFHWS2LS usa a versão Java especificada no parâmetro **JAVADIR**.

Sobre Esta Tarefa

Use o assistente XML do CICS para criar os mapeamentos de dados para o esquema XML.

Procedimento

1. Execute a tarefa em lote DFHSC2LS. O DFHSC2LS possui parâmetros opcionais que você pode selecionar para atender aos seus requisitos, tal como selecionar uma página de códigos específica ou especificar como manipular os dados de caracteres de comprimento variável. Use os parâmetros a seguir como um mínimo:
 - a. Especifique o local de seu arquivo de entrada no parâmetro **WSDL** ou **SCHEMA**. É possível usar um documento WSDL ou um esquema XML. Se seu arquivo de entrada contiver referências a outros esquemas ou documentos na Internet e o sistema usar um servidor proxy, especifique o nome de domínio ou o endereço IP e o número da porta do servidor proxy.
 - b. Especifique a linguagem de alto nível que você deseja gerar no parâmetro **LANG**. O assistente XML suporta linguagens COBOL, C, C++ e PL/I.

- c. Se você estiver implementando os mapeamentos de dados em um pacote configurável, especifique o nome e o local de um pacote configurável no parâmetro **BUNDLE**.

O assistente XML cria uma biblioteca de transformações suportadas na ligação XML. Para cada elemento global no arquivo de entrada, o assistente cria uma transformação separada.

Se você especificar o parâmetro **BUNDLE**, a tarefa em lote criará uma estrutura de diretório do pacote configurável no z/OS UNIX. O diretório do pacote configurável possui um subdiretório META-INF que contém o manifest do pacote configurável. A tarefa em lote também cria uma ligação XML no diretório de pacote configurável e coloca as estruturas de linguagem no local especificado. O assistente XML também coloca uma cópia do arquivo de entrada no diretório do pacote configurável. Se você não especificar o parâmetro **BUNDLE**, a tarefa em lote criará as estruturas de linguagem e a ligação XML somente no local especificado.

2. Instale o recurso BUNDLE. O recurso BUNDLE cria dinamicamente um recurso XMLTRANSFORM, que define o local do esquema XML ou documento WSDL, a ligação XML e as estruturas de linguagem.

Resultados

Quando você gera mapeamentos a partir de um esquema XML, o CICS gera uma estrutura de linguagem para cada elemento global que está presente no esquema.

Exemplo

O exemplo a seguir mostra o DFHSC2LS com o conjunto mínimo de parâmetros especificado.

```
//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
/*
```

O que Fazer Depois

Grave um programa de aplicativo para transformar os dados do aplicativo em XML ou vice-versa. É possível usar os mesmos mapeamentos para ambas as transformações.

Transformando dados do aplicativo em XML

É possível gravar um programa de aplicativo para transformar dados do aplicativo em XML.

Antes de Iniciar

Deve-se ter um recurso XMLTRANSFORM ativado que define a ligação XML e o esquema XML.

Sobre Esta Tarefa

O assistente XML gera os mapeamentos na ligação XML. Se você iniciou com uma estrutura de linguagem e usou DFHLS2SC, somente uma transformação para o XML é possível. Se você iniciou com um esquema XML, poderá haver muitas transformações de XML para a estrutura de linguagem, portanto, seu aplicativo deve selecionar qual elemento XML gerar.

Procedimento

1. O programa de aplicativo deve criar um canal e colocar os dados associados à estrutura de linguagem em um contêiner no modo de bit nesse canal.
2. Use o comando **TRANSFORM DATATOXML** API para transformar os dados em XML:

```
EXEC CICS TRANSFORM DATATOXML
XMLTRANSFORM('
  MyXmlTransformName
')
CHANNEL('
  MyChannelName
')
DATCONTAINER('
  SourceContainerName
')
XMLCONTAINER('
  TargetContainerName
')
```

Se o recurso XMLTRANSFORM suportar somente uma única transformação da estrutura de linguagem, você não precisará especificar o tipo de conversão no comando. Se várias transformações forem possíveis, inclua as opções a seguir em seu programa de aplicativo:

```
ELEMNAME(elementName) ELEMNAMELEN(elementNameLength)
```

Essas opções adicionais indicam o elemento XML para o qual os dados do aplicativo são transformados e colocados no contêiner de saída.

3. Instale o programa de aplicativo.

Resultados

Quando o aplicativo executa o comando **TRANSFORM DATATOXML**, o CICS verifica o recurso XMLTRANSFORM para localizar os mapeamentos na ligação XML e transforma os dados binários do aplicativo em XML usando os contêineres no canal. O XML é colocado no contêiner que o aplicativo especificou na opção XMLCONTAINER. O XML está em conformidade com o esquema XML que está definido no recurso XMLTRANSFORM.

O que Fazer Depois

Também é possível usar os mesmos mapeamentos para transformar XML em dados do aplicativo. Veja detalhes na seção “Transformando XML em dados do aplicativo” na página 407.

Transformando XML em dados do aplicativo

É possível gravar um programa de aplicativo para transformar XML em dados do aplicativo. Também é possível consultar o XML antes de transformá-lo.

Antes de Iniciar

Deve-se ter um recurso XMLTRANSFORM ativado que defina a ligação XML e o esquema XML.

Sobre Esta Tarefa

O assistente XML do CICS gera os mapeamentos na ligação XML. Se você iniciou com uma estrutura de linguagem e usou DFHLS2SC, é possível somente uma transformação a partir do XML. Se você iniciou com um esquema XML, pode haver muitas transformações de XML em estrutura de linguagem, portanto, seu aplicativo deve selecionar qual estrutura de linguagem usar como saída.

Procedimento

1. Crie um canal e coloque o XML em um contêiner no modo de texto nesse canal. Se o aplicativo colocar o XML em um contêiner no modo de BIT, o CICS tentará determinar a codificação dos dados de texto, mas pode levar mais tempo para processar o contêiner e a codificação poderá não estar correta.
2. Use o comando de API **TRANSFORM XMLTODATA**. Se apenas uma conversão for possível para o XML, será possível usar o comando a seguir:

```
EXEC CICS TRANSFORM XMLTODATA
XMLTRANSFORM('
  MyXmlTransformName
')
CHANNEL('
  MyChannelName
')
XMLCONTAINER('
  SourceContainerName
')
DATCONTAINER('
  TargetContainerName
')
```

Se mais de uma transformação for possível, use as opções adicionais a seguir:

```
EXEC CICS TRANSFORM XMLTODATA
XMLTRANSFORM('
  MyXmlTransformName
')
CHANNEL('
  MyChannelName
')
XMLCONTAINER('
  SourceContainerName
')
DATCONTAINER('
  TargetContainerName
')
ELEMNAME(
  elementName
) ELEMNAMELEN(
  elementNameLength
)
```

No segundo exemplo, o CICS retorna o nome do elemento que foi localizado na opção ELEMNAME. O aplicativo pode, então, usar o nome do elemento para determinar quais das bibliotecas de estruturas de linguagem usar para interpretar o conteúdo do contêiner de destino.

3. Instale o programa de aplicativo.

Resultados

Quando o aplicativo executa o comando **TRANSFORM XMLTODATA**, o CICS utiliza os detalhes no recurso XMLTRANSFORM para transformar o XML em dados binários do aplicativo usando os contêineres no canal.

Exemplo

O que Fazer Depois

Também é possível usar os mesmos mapeamentos para transformar dados do aplicativo em XML. Veja detalhes na seção “Transformando dados do aplicativo em XML” na página 405.

Mapeando e transformando dados do aplicativo e JSON

É possível gravar programas de aplicativo para transformar dados binários do aplicativo em JavaScript Object Notation (JSON) e vice-versa. O CICS suporta inúmeras linguagens de alto nível e fornece um assistente JSON para mapear como os dados são transformados durante o processamento de tempo de execução. O CICS usa a mesma tecnologia para mapear dados do aplicativo para mensagens JSON, como parte do suporte de serviços da web.

Antes de Iniciar

Deve-se ter Java instalado para executar o assistente JSON. As transformações podem ser realizadas internamente com o CICS ou usando um servidor JVM. Se você usar Java para as transformações, deverá ter um servidor JVM Axis2 instalado para transformar dados do aplicativo e JSON. Consulte Configurando um Servidor JVM para Axis2 para obter mais informações.

Sobre Esta Tarefa

A vantagem de usar esta abordagem para transformar dados do aplicativo para e a partir do JSON é que o CICS ultrapassa os recursos que são oferecidos por um analisador JSON. O CICS pode interpretar o JSON e executar conversões baseadas em registro dos dados do aplicativo. Portanto, é mais fácil e mais rápido para você criar aplicativos que trabalham com JSON usando essa abordagem.

O assistente CICS JSON é um utilitário fornecido que ajuda a criar os artefatos de mapeamento necessários para transformar dados binários do aplicativo em JSON ou para transformar JSON em dados binários do aplicativo. O assistente JSON cria os artefatos em um diretório do pacote configurável.

Procedimento

1. Crie um pacote configurável usando o assistente JSON. Esse pacote configurável contém os artefatos de mapeamento necessários para transformação de dados.
2. Instale o pacote configurável no CICS para disponibilizar os mapeamentos.

3. Crie ou atualize um programa de aplicativo para manipular a transformação de dados. Você possui duas opções:
 - Use os comandos da API **TRANSFORM DATATOJSON** e **TRANSFORM JSONTODATA** no programa de aplicativo. Essa é a abordagem recomendada.
 - Use o comando de API **LINK PROGRAM** para vincular-se ao DFHJSON de interface vinculável fornecido pelo CICS.

O aplicativo deve usar uma interface baseada em canal.

4. Execute o aplicativo para testar se a transformação funciona conforme o desejado.

Resultados

Seus dados do aplicativo são transformados em JSON ou seu JSON é transformado em dados do aplicativo.

O que Fazer Depois

As etapas 1 na página 408 a 4 são explicadas com mais detalhes nos tópicos a seguir.

O assistente JSON do CICS

O assistente CICS JSON é um conjunto de utilitários em lote que podem ajudá-lo a gerar os mapeamentos entre as estruturas de linguagem de alto nível e os esquemas JSON para executar transformações entre JSON e dados do aplicativo. O assistente suporta implementação rápida de aplicativos que executam processamento de JSON com quantidade mínima de esforço de programação.

O uso do assistente JSON para CICS reduz a quantidade de código que você deve gravar para analisar ou construir JSON; o CICS transforma dados entre fragmentos JSON e a estrutura de dados de um programa aplicativo.

O assistente JSON pode criar um esquema JSON a partir de uma estrutura de linguagem simples, ou uma estrutura de linguagem a partir de um esquema XML existente, e suporta COBOL, C/C++ e PL/I. Ele também gera metadados que o CICS usa no tempo de execução para converter automaticamente dados XML em dados do aplicativo binários ou vice-versa, os metadados são definidos em uma ligação XML e armazenados no z/OS UNIX. O esquema para a ligação XML está no diretório `/usr/lpp/cicsts/cicsts52/schemas/xmltransform/` no z/OS UNIX.

Os programas utilitários DFHJS2LS e DFHLS2JS são conhecidos coletivamente como o assistente CICS JSON:

DFHLS2JS

DFHLS2JS mapeia as estruturas de linguagem de alto nível para esquemas JSON.

DFHJS2LS

DFHJS2LS mapeia esquemas JSON para estruturas de linguagem de alto nível.

Os procedimentos JCL para executar ambos os programas estão na biblioteca `hlq.XDFHINST` em que `hlq` é o qualificador de alto nível de sua instalação do CICS.

O modo de uso relevante para o procedimento DFHLS2JS ou DFHJS2LS depende de seus requisitos:

- DFHLS2JS: linguagem de alto nível para conversão de esquema JSON para a interface vinculável
- DFHJS2LS: esquema JSON para conversão de linguagem de alto nível para interface vinculável
- DFHLS2JS: linguagem de alto nível para conversão de esquema JSON para serviços de solicitação-resposta
- DFHJS2LS: esquema JSON para conversão de linguagem de alto nível para serviços de solicitação-resposta
- DFHJS2LS: Esquema JSON para conversão de linguagem de alto nível para serviços RESTful

Os dois mapeamentos não são simétricos:

- Se você processar uma estrutura de dados de linguagem com DFHLS2JS e, em seguida, processar o esquema JSON resultante com o DFHJS2LS, não espere que a estrutura de dados final seja a mesma que a original.
- Se você processar um esquema JSON com DFHJS2LS e, em seguida, processar a estrutura de linguagem resultante com o DFHLS2JS, não espere que o esquema JSON final seja o mesmo que o original.
- Em alguns casos, DFHJS2LS gera estruturas de linguagem que não são suportadas pelo DFHLS2JS.

Você deve codificar estruturas de linguagem de alto nível que são processadas pelo DFHLS2JS de acordo com as regras da linguagem, conforme implementado nos compiladores de linguagem que o CICS suporta.

DFHLS2JS: Linguagem de alto nível para conversão de esquema JSON para a interface vinculável

O procedimento catalogado do DFHLS2JS gera um esquema JSON e um arquivo de ligação JSON a partir de uma estrutura de linguagem de alto nível. Use o DFHLS2JS quando desejar criar um programa CICS que possa analisar ou criar JSON. Descrições das instruções de controle de tarefa, parâmetros simbólicos, os parâmetros de entrada para DFHLS2JS e uma tarefa de exemplo que utiliza DFHLS2JS são fornecidos para ajudá-lo a utilizar este procedimento.

O procedimento JCL DFHLS2JS é instalado no conjunto de dados *HLQ* .XDFHINST , em que *HLQ* é o qualificador de alto nível no qual o CICS está instalado.

Instruções de controle da tarefa para DFHLS2JS

JOB Inicia a tarefa.

EXEC Especifica o nome do procedimento (DFHLS2JS).

INPUT.SYSUT1 DD

Especifica a entrada. Os parâmetros de entrada são especificados no fluxo de entrada. No entanto, também é possível defini-los em um conjunto de dados ou em um membro de um conjunto de dados particionados.

Parâmetros simbólicos

Os parâmetros simbólicos a seguir são definidos no DFHLS2JS:

JAVADIR = *path*

Especifica o nome do diretório Java que é usado pelo DFHLS2JS. O valor desse parâmetro é anexado a */usr/lpp/* fornecendo um nome de caminho completo de */usr/lpp/path*.

Normalmente, você não especifica este parâmetro. O valor padrão é o valor que foi fornecido à tarefa de instalação do CICS (DFHISTAR) no parâmetro **JAVADIR**.

PATHPREF = *prefix*

Especifica um prefixo opcional que estende o caminho do diretório do z/OS UNIX que é usado em outros parâmetros. O padrão é a cadeia vazia.

Normalmente, você não especifica este parâmetro. O valor padrão é o valor que foi fornecido para a tarefa de instalação do CICS (DFHISTAR) no parâmetro **PATHPREF**.

SERVICE = *value*

Use este parâmetro somente quando orientado a fazer isso pelo suporte IBM.

TMPDIR = *tmpdir*

Especifica o local de um diretório no z/OS UNIX que o DFHLS2JS usa como uma área de trabalho provisória. O ID do usuário sob o qual a tarefa é executada deve ter permissão de leitura e gravação para esse diretório.

O valor padrão é /tmp.

TMPFILE = *tmpprefix*

Especifica um prefixo que o DFHLS2JS utiliza para construir os nomes dos arquivos de área de trabalho provisória.

O valor padrão é LS2JS.

USSDIR = *path*

Especifica o nome do diretório do CICS TS no sistema de arquivos do z/OS UNIX. O valor desse parâmetro é anexado a /usr/lpp/cicsts/ fornecendo um nome de caminho completo de /usr/lpp/cicsts/*path* . Isso deve ser especificado como '.' (ponto) se o padrão for usado.

Normalmente, você não especifica este parâmetro. O valor padrão é o valor que foi fornecido à tarefa de instalação do CICS (DFHISTAR) no parâmetro **USSDIR**.

A área de trabalho provisória

DFHLS2JS cria os três arquivos temporários a seguir no tempo de execução:

```
tmpdir / tmpprefix .in  
tmpdir / tmpprefix .out  
tmpdir / tmpprefix .err
```

em que:

tmpdir é o valor especificado no parâmetro **TMPDIR**.

tmpprefix é o valor especificado no parâmetro **TMPFILE**.

Os nomes padrão para os arquivos (quando **TMPDIR** e **TMPFILE** não são especificados) são conforme a seguir:

```
/tmp/LS2JS.in  
/tmp/LS2JS.out  
/tmp/LS2JS.err
```

Importante: O DFHLS2JS não bloqueia o acesso aos arquivos do z/OS UNIX ou aos membros do conjunto de dados. Portanto, se duas ou mais instâncias de DFHLS2JS forem executadas simultaneamente e usarem os mesmos arquivos da

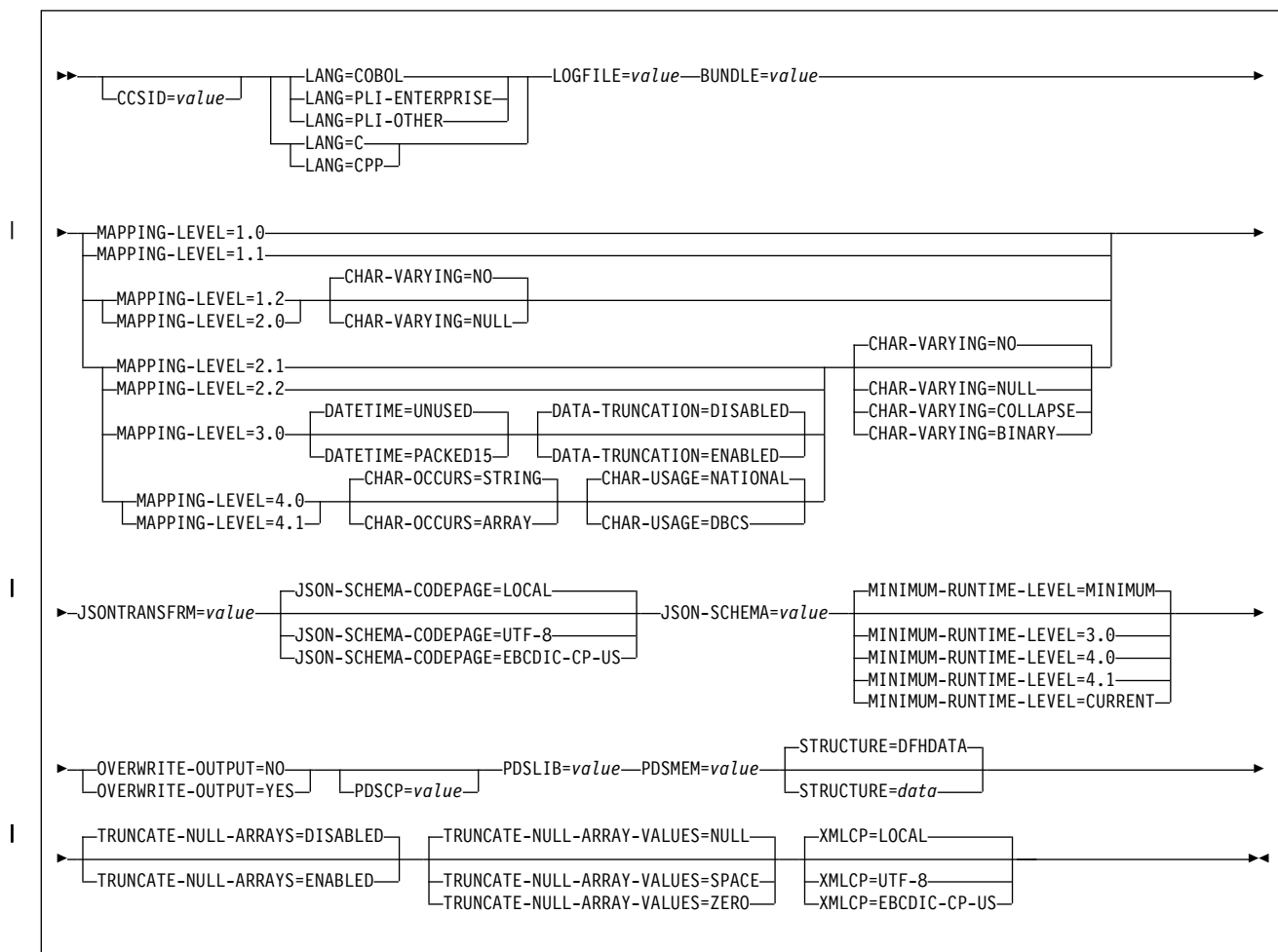
área de trabalho provisória, nada impede que uma tarefa sobrescreva os arquivos da área de trabalho enquanto outra tarefa está usando-os, levando a falhas imprevisíveis.

Crie uma convenção de nomenclatura e procedimentos operacionais que evitem esta situação; por exemplo, é possível usar o parâmetro simbólico do sistema **SYSUID** para gerar nomes de arquivos da área de trabalho que são exclusivos para um usuário individual.

Estes arquivos temporários são excluídos antes do encerramento da tarefa.

Parâmetros de entrada para DFHLS2JS

O diagrama de sintaxe a seguir mostra os parâmetros de entrada disponíveis:



Utilização de Parâmetros

Os parâmetros devem estar em conformidade com as regras a seguir:

- Você pode especificar os parâmetros de entrada em qualquer ordem.
- Cada parâmetro deve ser iniciado em uma nova linha.
- Um parâmetro (e seu caractere de continuação, se você usar um) não deve estender além da coluna 72; as colunas 73 a 80 devem conter espaços em branco.

- Se um parâmetro é muito longo para se ajustar em uma única linha, use um caractere de asterisco (*) no final da linha para indicar que o parâmetro continua na próxima linha. Tudo (incluindo espaços) antes do asterisco é considerado parte do parâmetro.
- Um caractere de sinal de número (#) na primeira posição do caractere da linha é um caractere de comentário. A linha é ignorada.
- Uma vírgula na posição de último caractere da linha é um separador de linha opcional e é ignorado.

Descrições de Parâmetros

BUNDLE = *value*

Especifica o caminho e o nome do diretório do pacote configurável no z/OS UNIX. Se você especificar esse valor, o assistente JSON gerará um pacote configurável contendo a ligação JSON. As informações de caminho para esse parâmetro substitui quaisquer informações de caminho para o parâmetro **JSONTRANSFRM**.

Opcionalmente, é possível especificar um archive em vez de um nome de diretório. O assistente JSON suporta archives .zip e .jar. No entanto, deve-se descompactar o archive antes de tentar instalar o recurso BUNDLE.

CCSID = *value*

Especifica o CCSID que é usado no tempo de execução para codificar dados de caractere na estrutura de dados do aplicativo. O valor desse parâmetro substitui o valor do parâmetro de inicialização do sistema **LOCALCCSID**. O *value* deve ser um CCSID EBCDIC que é suportado pelos serviços de conversão Java e z/OS. Se você não especificar esse parâmetro, a estrutura de dados do aplicativo será codificada usando o CCSID especificado no parâmetro de inicialização do sistema.

Este parâmetro pode ser utilizado com qualquer nível de mapeamento.

CHAR-VARYING = { **NO** | **NULL** | **COLLAPSE** | **BINARY** }

Especifica como os campos de caractere na estrutura de linguagem são mapeados quando o nível de mapeamento é 1.2 ou superior. Um campo de caractere em COBOL é uma cláusula Picture do tipo X; por exemplo, PIC(X) 10. Um campo de caractere em C/C++ é uma matriz de caracteres. Este parâmetro não se aplica a estruturas de linguagem PL/I Enterprise ou Other. Você pode selecionar estas opções:

NO Os campos de caractere são mapeados para uma sequência JSON e são processados como campos de comprimento fixo. O comprimento máximo dos dados é igual ao comprimento do campo. NO é o valor padrão para o parâmetro **CHAR-VARYING** para COBOL e PL/I nos níveis de mapeamento 2.0 e anteriores.

NULL Os campos de caractere são mapeados para uma sequência JSON e são processados como sequências com terminação nula. O CICS inclui um caractere nulo de terminação ao transformar a partir de um esquema JSON. O comprimento máximo da sequência de caracteres é calculado como um caractere menor que o comprimento indicado na estrutura de linguagem. NULL é o valor padrão para o parâmetro **CHAR-VARYING** para C/C++.

COLLAPSE

Os campos de caractere são mapeados para uma sequência JSON. O espaço em branco final neste campo não é incluído no esquema JSON.

COLLAPSE é o valor padrão para o parâmetro **CHAR-VARYING** para COBOL e PL/I no nível de mapeamento 2.1 em diante.

BINARY

Os campos de caractere são mapeados para uma sequência JSON contendo dados codificados em base64 e são processados como campos de comprimento fixo. O valor BINARY no parâmetro **CHAR-VARYING** está disponível somente nos níveis de mapeamento 2.1 e superiores.

CHAR-OCCURS = { STRING | **ARRAY** }

Especifica como matrizes de caracteres na estrutura de linguagem são mapeadas quando o nível de mapeamento é 4.0 ou superior. Por exemplo, PIC X OCCURS 20. Este parâmetro destina-se ao uso somente pela linguagem COBOL.

ARRAY

As matrizes de caracteres são mapeadas para uma matriz JSON. Isso significa que cada caractere é mapeado como um elemento JSON individual. Este também é o comportamento nos níveis de mapeamento 3.0 e anteriores.

CADEIA

Matrizes de caracteres são mapeadas para uma sequência JSON. Isso significa que a matriz COBOL inteira é mapeada como um único elemento JSON.

CHAR-USAGE = { NATIONAL | **DBCS** }

Em COBOL, o tipo de dados nacional, PIC N, pode ser usado para dados UTF-16 ou DBCS. Essa configuração é controlada pela opção do compilador NSYMBOL. Deve-se configurar o parâmetro **CHAR-USAGE** no assistente com o mesmo valor que a opção do compilador NSYMBOL para assegurar que os dados sejam manipulados apropriadamente. Isto é configurado geralmente como CHAR-USAGE=NATIONAL quando você usa UTF-16.

DBCS Dados dos campos PIC (*n*) são tratados como dados codificados em UTF-16.

NATIONAL

Dados dos campos PIC (*n*) são tratados como dados codificados em DBCS.

DATA-TRUNCATION = { DISABLED | **ENABLED** }

Especifica se dados de comprimento variável são tolerados em uma estrutura de campo de comprimento fixo:

DISABLED

Se os dados forem menores do que o comprimento fixo que o CICS está esperando, o CICS rejeitará os dados truncados e emitirá uma mensagem de erro.

ENABLED

Se os dados são menores do que o comprimento fixo que o CICS está esperando, o CICS tolera os dados truncados e processa os dados ausentes como valores nulos.

DATETIME = { UNUSED | **PACKED15** }

Especifica se propriedades de data/hora JSON na estrutura de linguagem de alto nível, incluindo valores do CICS ABSTIME, são mapeados como registros de data e hora:

PACKED15

Quaisquer propriedades de data/hora JSON são mapeadas como registros de data e hora.

UNUSED

Quaisquer propriedades de data/hora JSON não são mapeadas como registros de data e hora. Esse mapeamento é o padrão.

Você pode configurar este parâmetro no nível de mapeamento 3.0.

JSON-SCHEMA = *value*

O nome completo do arquivo z/OS UNIX no qual o esquema JSON é gravado.

JSON-SCHEMA-CODEPAGE = { **LOCAL** | **UTF-8** | **EBCDIC-CP-US** }

Especifica a página de códigos que é usada para gerar os documentos do Esquema JSON.

LOCAL

Especifica que os Esquemas JSON são gerados usando a página de códigos padrão para o sistema de arquivos.

UTF-8 Especifica que os Esquemas JSON são gerados usando a página de códigos UTF-8.

EBCDIC-CP-US

Especifica que os Esquemas JSON são gerados usando a página de códigos US EBCDIC.

JSONTRANSFRM = *value*

Este parâmetro é obrigatório para o modo LINKable (vinculável), mas inválido para o modo de solicitação-resposta. Ele indica o nome que é usado para o recurso de pacote configurável **JSONTRANSFRM** no CICS.

LANG = **COBOL**

Especifica que a linguagem de programação da estrutura de linguagem de alto nível é COBOL.

LANG = **PLI-ENTERPRISE**

Especifica que a linguagem de programação da estrutura de linguagem de alto nível é Enterprise PL/I.

LANG = **PLI-OTHER**

Especifica que a linguagem de programação da estrutura de linguagem de alto nível é um nível de PL/I diferente de Enterprise PL/I.

LANG = **C**

Especifica que a linguagem de programação da estrutura de linguagem de alto nível é C.

LANG = **CPP**

Especifica que a linguagem de programação da estrutura de linguagem de alto nível é C++.

LOGFILE = *value*

O nome completo do arquivo z/OS UNIX no qual o DFHLS2JS grava seu log de atividades e informações de rastreamento. Se ele não existir, o DFHLS2JS criará o arquivo, mas não a estrutura de diretório.

Pode ser solicitado que você use este parâmetro pela organização de serviço da IBM caso encontre problemas com o DFHLS2JS.

MAPPING-LEVEL = { 1.0 | 1.1 | 1.2 | 2.0 | 2.1 | 2.2 | 3.0 | 4.0 | 4.1 }

Especifica o nível de mapeamento para o assistente usar ao gerar a ligação JSON e as estruturas de linguagem. Deve-se usar o nível de mapeamento 3.0 ou superior.

MINIMUM-RUNTIME-LEVEL = { MINIMUM | 3.0 | 4.0 | | 4.1 CURRENT }

Especifica o ambiente de tempo de execução mínimo do CICS no qual a ligação JSON pode ser implementada. Se você selecionar um nível que não corresponde ao outros parâmetros especificados, receberá uma mensagem de erro. As opções que podem ser selecionadas são as seguintes:

MINIMUM

O menor nível de tempo de execução possível do CICS é alocado automaticamente com base nos parâmetros que você especificou.

3,0 Especifique o nível de tempo de execução 3.0 ou acima se desejar usar o assistente CICS JSON e tirar vantagem dos mapeamentos de dados avançados.

4.0 O arquivo de ligação de serviço da web gerado é implementado com sucesso em um CICS TS 5.2 ou em uma região posterior. Com este nível de tempo de execução, é possível usar um nível de mapeamento 4.0 ou anterior para o parâmetro **MAPPING-LEVEL**. É possível utilizar qualquer parâmetro opcional neste nível.

4.1 O arquivo de ligação de serviço da web gerado é implementado com sucesso em uma região do CICS TS 5.2 que tenha o APAR PI67641 aplicado ou em qualquer região do CICS TS 5.3 ou posterior. Com este nível de tempo de execução, é possível utilizar um nível de mapeamento 4.1 ou anterior para o parâmetro **MAPPING-LEVEL**.

CURRENT

Use este nível de tempo de execução para implementar o arquivo de ligação gerado em uma região CICS que possui o ambiente de tempo de execução que é configurado com aquele utilizado para gerar o arquivo de ligação.

OVERWRITE-OUTPUT = { NO | YES }

Controla se **BUNDLES** existentes do CICS no sistema de arquivos podem ser sobrescritos.

NO Qualquer **BUNDLE** existente não é substituído. Se um **BUNDLE** existente for localizado, o DFHLS2JS emitirá a mensagem de erro DFHPI9689E e finalizará.

YES Qualquer **BUNDLE** existente é substituído. Se um **BUNDLE** existente for localizado, a mensagem DFHPI9683W será emitida para informá-lo que o arquivo foi substituído.

PDSCP = value

Especifica a página de códigos que é usada nos membros do conjunto de dados particionados, em que *value* é um número de CCSID ou um número da página de códigos Java. Se você não especificar esse parâmetro, a página de códigos do z/OS UNIX System Services será usada. Por exemplo, você pode especificar PDSCP=037.

PDSLIB = value

Especifica o nome do conjunto de dados particionados que contém as estruturas de dados de linguagem de alto nível a serem processadas.

Restrição: Os registros no conjunto de dados particionados devem ter um comprimento fixo de 80 bytes.

PDSMEM = *value*

Especifica o nome do membro do conjunto de dados particionados que contém as estruturas de linguagem de alto nível que você deseja processar.

STRUCTURE = { DFHDATA | *data* }

O nome da estrutura de dados de nível superior em C e C++. O padrão é DFHDATA.

TRUNCATE-NULL-ARRAYS = { DISABLED | **ENABLED** }

Especifica como matrizes estruturadas são processadas no nível de mapeamento 4.1 ou superior. Se ativado, o CICS tentará reconhecer registros vazios em uma matriz (consulte TRUNCATE-NULL-ARRAY-VALUES para obter mais informações sobre como identificar os registros vazios). Se cinco registros de matriz vazia consecutivos forem detectados, a matriz será truncada no primeiro registro ao gerar XML/JSON. Esse recurso de truncamento é ativado somente para matrizes com conteúdo estruturado, matrizes de campos primitivos simples não estão sujeitas a truncamento. O truncamento de matrizes pode resultar em uma representação mais concisa dos dados em JSON/XML, mas não é sem risco. Se cinco registros de dados consecutivos forem identificados incorretamente como armazenamento não inicializado (talvez porque eles legitimamente contêm valores baixos), poderá haver perda de dados. Se TRUNCATE-NULL-ARRAYS estiver ativado e TRUNCATE-NULL-ARRAY-VALUES não estiver configurado, o valor padrão para TRUNCATE-NULL-ARRAY-VALUES será usado.

TRUNCATE-NULL-ARRAY-VALUES = { NULL | **SPACE** | **ZERO** }

Especifica quais valores são tratados como vazios para processamento de TRUNCATE-NULL-ARRAYS no nível de mapeamento 4.1 ou superior. Por padrão, o valor nulo (0x00 ou valores baixos) é tratado como vazio. Se todos os bytes de armazenamento em um registro de uma matriz estruturada contêm valores nulos, o registro inteiro é considerado vazio. Um ou mais dos valores NULL, SPACE e ZERO podem ser especificados em uma lista separada por vírgula, em que NULL indica um caractere nulo (0x00), SPACE indica um espaço SBCS EBCDIC (0x40) e ZERO indica um zero decimal zonado não sinalizado (0xF0). Qualquer combinação correspondente dos bytes selecionados em um registro de matriz estruturada fará com que o registro inteiro seja identificado como vazio. Se TRUNCATE-NULL-ARRAY-VALUES tem um valor definido, TRUNCATE-NULL-ARRAYS deve ser ativado.

```
//LS2JS JOB '  
accounting information  
,  
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: Esquema JSON para conversão de linguagem de alto nível para interface vinculável

O procedimento catalogado DFHJS2LS gera uma estrutura de dados de linguagem de alto nível e uma ligação JSON a partir de um esquema JSON. Use DFHJS2LS quando desejar criar um programa CICS que possa analisar ou criar JSON. Este tópico lista as instruções de controle de tarefa, parâmetros simbólicos, parâmetros de entrada e suas descrições para DFHJS2LS.

O procedimento JCL DFHJS2LS é instalado no conjunto de dados *HLQ.XDFHINST*, em que *HLQ* é o qualificador de alto nível no qual o CICS está instalado.

Instruções de controle da tarefa para DFHJS2LS

JOB Inicia a tarefa.

EXEC Especifica o nome do procedimento (DFHJS2LS).

INPUT.SYSUT1 DD

Especifica a entrada. Os parâmetros de entrada são especificados no fluxo de entrada. Também é possível defini-los em um conjunto de dados ou em um membro de um conjunto de dados particionados.

Parâmetros simbólicos

Os parâmetros simbólicos a seguir são definidos no DFHJS2LS:

JAVADIR = *path*

Especifica o nome do diretório Java que é usado pelo DFHJS2LS. O valor desse parâmetro é anexado a */usr/lpp/* fornecendo um nome de caminho completo de */usr/lpp/path*.

Normalmente, você não especifica este parâmetro. O valor padrão é o valor que foi fornecido à tarefa de instalação do CICS (DFHISTAR) no parâmetro **JAVADIR**.

PATHPREF = *prefix*

Especifica um prefixo opcional que estende o caminho do diretório do z/OS UNIX que é usado em outros parâmetros. O padrão é a cadeia vazia.

Normalmente, você não especifica este parâmetro. O valor padrão é o valor que foi fornecido para a tarefa de instalação do CICS (DFHISTAR) no parâmetro **PATHPREF**.

SERVICE = *value*

Use este parâmetro somente quando orientado a fazer isso pelo suporte IBM.

TMPDIR = *tmpdir*

Especifica o local de um diretório no z/OS UNIX que o DFHJS2LS utiliza como uma área de trabalho provisória. O ID do usuário sob o qual a tarefa é executada deve ter permissão de leitura e gravação para esse diretório.

O valor padrão é */tmp*.

TMPFILE = *tmpprefix*

Especifica um prefixo que o DFHJS2LS usa para construir os nomes dos arquivos de área de trabalho provisória.

O valor padrão é JS2LS.

USSDIR = *path*

Especifica o nome do diretório CICS TS no sistema de arquivos do UNIX System Services. O valor desse parâmetro é anexado a */usr/lpp/cicsts/* para

produzir um nome do caminho completo de `/usr/lpp/cicsts/ path` . Isso deve ser especificado como `'.'` (ponto) se o padrão for usado.

Normalmente, você não especifica este parâmetro. O valor padrão é o valor que foi fornecido para a tarefa de instalação do CICS (DFHISTAR) no parâmetro **USSDIR**.

A área de trabalho provisória

DFHJS2LS cria os três arquivos temporários a seguir no tempo de execução:

```
tmpdir / tmpprefix .in  
tmpdir / tmpprefix .out  
tmpdir / tmpprefix .err
```

em que:

tmpdir é o valor especificado no parâmetro **TMPDIR**.

tmpprefix é o valor especificado no parâmetro **TMPFILE**.

Os nomes padrão para os arquivos (quando **TMPDIR** e **TMPFILE** não são especificados) são conforme a seguir:

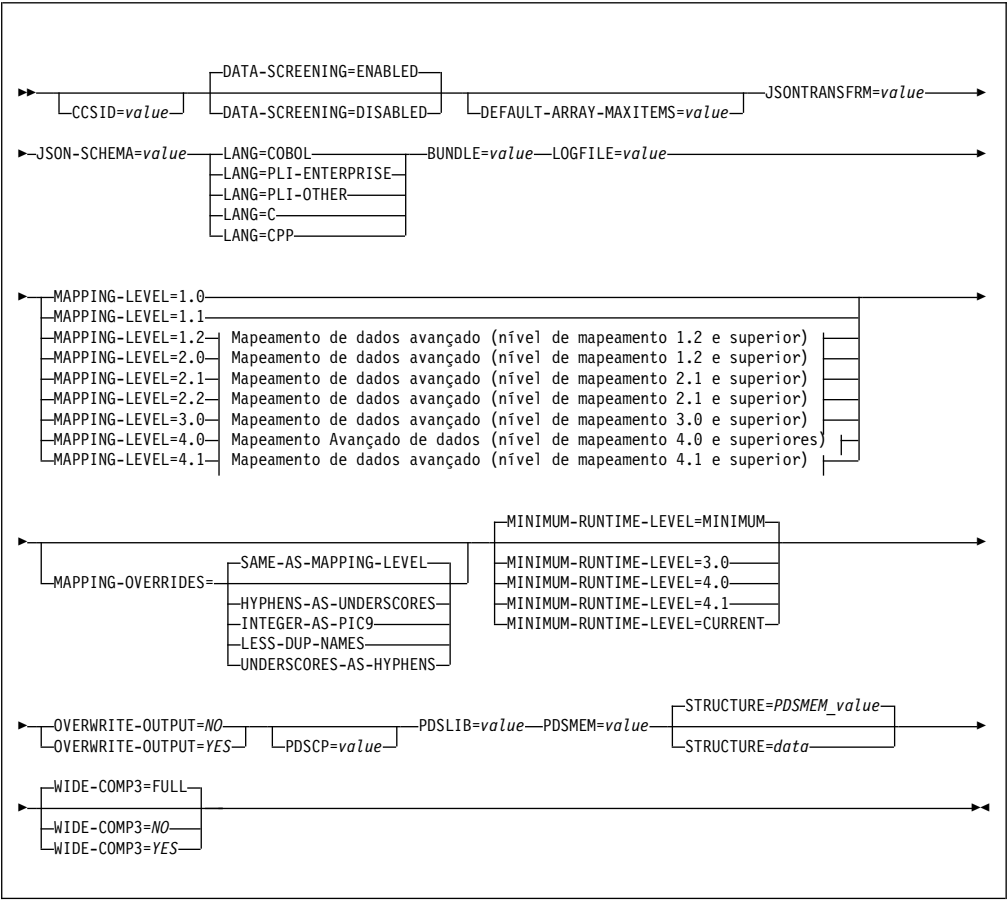
```
/tmp/JS2LS.in  
/tmp/JS2LS.out  
/tmp/JS2LS.err
```

Importante: O DFHJS2LS não bloqueia o acesso aos arquivos do z/OS UNIX ou aos membros do conjunto de dados. Portanto, se duas ou mais instâncias de DFHJS2LS são executadas simultaneamente e usam os mesmos arquivos da área de trabalho provisória, nada impede que uma tarefa sobrescreva os arquivos da área de trabalho enquanto outra tarefa está usando-os, levando a falhas imprevisíveis.

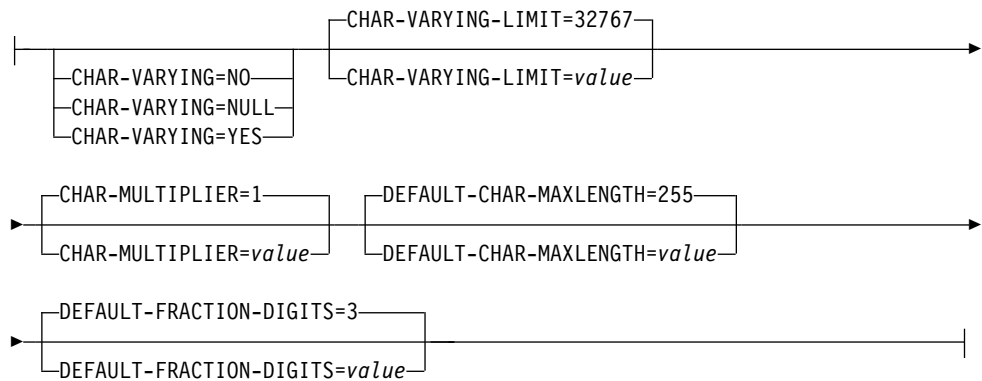
Portanto, você será avisado para planejar uma convenção de nomenclatura e procedimentos operacionais que evitem esta situação; por exemplo, é possível usar o parâmetro simbólico do sistema **SYSUID** para gerar nomes de arquivos da área de trabalho que sejam exclusivos para um usuário individual.

Estes arquivos temporários são excluídos antes do encerramento da tarefa.

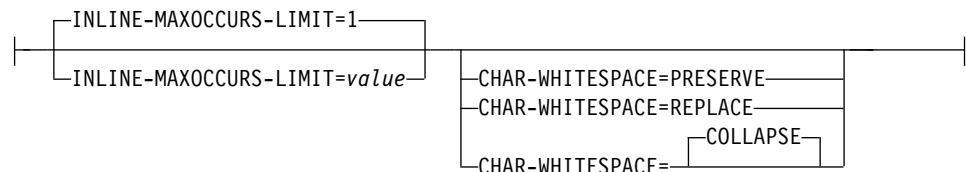
Parâmetros de entrada para DFHJS2LS



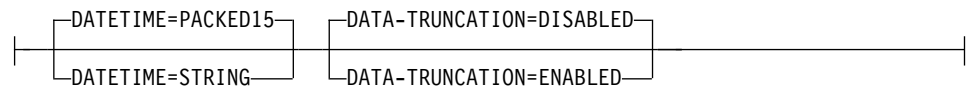
Mapeamento de Dados Avançado (Nível de Mapeamento 1.2 e Superior):



Mapeamento de Dados Avançado (Nível de Mapeamento 2.1 e Superior):



Mapeamento de dados avançado (nível de mapeamento 3.0 e superior):



Utilização de Parâmetros

- Você pode especificar os parâmetros de entrada em qualquer ordem.
- Cada parâmetro deve ser iniciado em uma nova linha.
- Um parâmetro (e seu caractere de continuação, se você usar um) não deve estender além da coluna 72; as colunas 73 a 80 devem conter espaços em branco.
- Se um parâmetro é muito longo para se ajustar em uma única linha, use um caractere de asterisco (*) no final da linha para indicar que o parâmetro continua na próxima linha. Tudo (incluindo espaços) antes do asterisco é considerado parte do parâmetro.
- Um caractere # na primeira posição de caractere da linha é um caractere de comentário. A linha é ignorada.
- Uma vírgula na posição de último caractere da linha é um separador de linha opcional e é ignorado.

Descrições de Parâmetros

BUNDLE = *value*

Especifica o caminho e o nome do diretório do pacote configurável no z/OS UNIX. Se você especificar esse valor, o assistente JSON gerará a ligação JSON no diretório de pacote configurável e criará um manifest do pacote

configurável para você. As informações de caminho para esse parâmetro substitui qualquer informação de caminho para o parâmetro **JSONTRANSFRM**.

CCSID = *value*

Especifica o CCSID que é usado no tempo de execução para codificar dados de caractere na estrutura de dados do aplicativo. O valor desse parâmetro substitui o valor do parâmetro de inicialização do sistema **LOCALCCSID**. O *value* deve ser um CCSID EBCDIC que seja suportado pelos serviços de conversão Java e z/OS. Se você não especificar esse parâmetro, a estrutura de dados do aplicativo será codificada usando o CCSID especificado no parâmetro de inicialização do sistema.

Este parâmetro pode ser utilizado com qualquer nível de mapeamento.

CHAR-MULTIPLIER = { 1 | *value* }

Especifica o número de bytes para permitir cada caractere quando o nível de mapeamento é 1.2 ou posterior. O *value* deste parâmetro pode ser um número inteiro positivo no intervalo de 1 a 2.147.483.647. Todos os mapeamentos baseados em caracteres não numéricos estão sujeitos a este multiplicador. Campos binários, numéricos, zoneados e decimais compactados não estão sujeitos a este multiplicador.

Este parâmetro pode ser útil se, por exemplo, você está planejando usar caracteres DBCS onde pode optar por um multiplicador de 3 para permitir espaço para possíveis caracteres shift-out e shift-in em cada caractere de byte duplo no tempo de execução.

Quando você configura **CCSID=1200** (indicando UTF-16), os únicos valores válidos para **CHAR-MULTIPLIER** são 2 ou 4. Ao usar UTF-16, o valor padrão é 2. Use **CHAR-MULTIPLIER=2** quando esperar que os dados do aplicativo contenham caracteres que requerem 1 unidade de codificação UTF-16. Use **CHAR-MULTIPLIER=4** quando esperar que os dados do aplicativo contenham caracteres que requerem 2 unidades de codificação UTF-16.

Nota: A configuração de **CHAR-MULTIPLIER** como 1 não impede o uso de caracteres DBCS e sua configuração como 2 não impede o uso de pares substitutos UTF-16. No entanto, se caracteres largos forem usados rotineiramente, alguns valores válidos não se ajustarão no campo alocado. Se um valor **CHAR-MULTIPLIER** maior for usado, poderá ser possível armazenar mais caracteres no campo alocado que são válidos no XML. Deve-se tomar cuidado quanto à conformidade com as restrições de intervalo apropriadas.

CHAR-VARYING = { **NO** | **NULL** | **YES** }

Especifica como os dados de caracteres de comprimento variável são mapeados quando o nível de mapeamento é 1.2 ou superior. Os tipos de dados binários de comprimento variável são sempre mapeados para um contêiner ou uma estrutura variável. Se você não especificar esse parâmetro, o mapeamento padrão dependerá da linguagem especificada. Você pode selecionar estas opções:

NO Os dados de caractere de comprimento variável são mapeados como sequências de comprimento fixo.

NULL Os dados de caractere de comprimento variável são mapeados para sequências com terminação nula.

YES Dados de caractere de comprimento variável são mapeados para um tipo de dados CHAR VARYING em PL/I. Nas linguagens COBOL, C e C++, os dados de caracteres de comprimento variável são mapeados

para uma representação equivalente que consiste em dois elementos relacionados: o comprimento de dados e os dados.

CHAR-VARYING-LIMIT = { 32767 | *value* }

Especifica o tamanho máximo de dados binários e de dados de caractere de comprimento variável que são mapeados para a estrutura de linguagem quando o nível de mapeamento é 1.2 ou superior. Se os dados de caractere ou binários forem maiores do que o valor especificado neste parâmetro, eles serão mapeados para um contêiner e o nome do contêiner será usado na estrutura de linguagem gerada. O *valor* pode variar de 0 ao padrão de 32 767 bytes.

CHAR-WHITESPACE = COLLAPSE | REPLACE | PRESERVE

Especifica como o espaço em branco nos valores de sequência de tipos será manipulado pelo CICS.

COLLAPSE

Espaços em branco iniciais, finais e integrados são removidos e todas as guias, novas linhas e espaços consecutivos são substituídos por caracteres de espaço único.

TROCAR

As guias ou novas linhas são substituídas pelo número apropriado de espaços.

PRESERVE

Retém qualquer espaço em branco no valor dos dados.

Se o parâmetro **CHAR-WHITESPACE** não for configurado, o espaço em branco será reduzido.

Nota: Este parâmetro não se aplica aos campos com um formato de data/hora , URI , base64Binary ou hexBinary , em que o espaço em branco é sempre reduzido.

DATA-SCREENING = { ENABLED | DISABLED }

Especifica se os dados fornecidos pelo aplicativo foram verificados com relação aos erros.

ENABLED

Quaisquer dados inconsistentes de tempo de execução fornecidos pelo aplicativo com a estrutura de linguagem são tratados como um erro e a mensagem DFHPI1010 é emitida. Uma resposta de erro é retornada ao aplicativo.

DISABLED

Os valores dos dados de tempo de execução fornecidos pelo aplicativo que são inconsistentes com a estrutura de linguagem são substituídos pelos valores padrão. Por exemplo, um zero substitui um valor inválido em um campo numérico. A mensagem DFHPI1010 não é emitida e uma resposta normal é retornada ao aplicativo. Esse recurso pode ser usado para evitar as respostas de erro INVALID_PACKED_DEC e INVALID_ZONED_DEC que são geradas a partir de campos de saída não inicializados.

DATA-TRUNCATION = { DISABLED | ENABLED }

Especifica se dados de comprimento variável são tolerados em uma estrutura de campo de comprimento fixo:

DISABLED

Se os dados forem menores do que o comprimento fixo que o CICS está esperando, o CICS rejeitará os dados truncados e emitirá uma mensagem de erro.

ENABLED

Se os dados são menores do que o comprimento fixo que o CICS está esperando, o CICS tolera os dados truncados e processa os dados ausentes como valores nulos.

DATETIME = { **PACKED15** | **STRING** }

Especifica que propriedades de data/hora JSON são mapeadas para o formato de dados CICS ABSTIME ou para texto:

PACKED15

Os campos de propriedades de data/hora JSON são mapeados para o formato CICS ABSTIME.

CADEIA

Propriedades de data/hora JSON são mapeadas para texto. Esse mapeamento é o mesmo que todos os níveis de mapeamentos anteriores.

Você pode usar este parâmetro no nível de mapeamento 3.0.

DEFAULT-ARRAY-MAXITEMS = *value*

Especifica o limite máximo da matriz a ser aplicado quando nenhuma informação de ocorrência máxima (maxItems) estiver implícita no esquema JSON. Se esse parâmetro não estiver configurado, nenhum limite máximo será aplicado. O valor desse parâmetro pode ser um número inteiro positivo no intervalo de 1 a 2147483647. Esse parâmetro pode ser combinado com o uso do parâmetro **INLINE-MAXOCCURS-LIMIT** para influenciar como as matrizes JSON são mapeadas para as estruturas de linguagem.

DEFAULT-CHAR-MAXLENGTH = { **255** | *value* }

Especifica o comprimento da matriz padrão de dados de caractere em caracteres para mapeamentos, em que nenhum comprimento está implícito no documento de esquema JSON, quando o nível de mapeamento é 1.2 ou posterior. O *value* deste parâmetro pode ser um número inteiro positivo no intervalo de 1 a 2147483647.

DEFAULT-FRACTION-DIGITS = { **3** | *value* }

Especifica o número padrão de dígitos fracionários para uso em um tipo de esquema decimal XML. O padrão é 3. Para COBOL, o intervalo válido é de 0 a 17 ou de 0 a 30 se o parâmetro **WIDE-COMP3** está sendo usado. Para C ou PLI, o intervalo válido é de 0 a 30.

INLINE-MAXOCCURS-LIMIT = { **1** | *value* }

Especifica se o conteúdo de repetição de variável sequencial é usado com base no atributo maxItems das palavras-chave do esquema JSON.

O parâmetro **INLINE-MAXOCCURS-LIMIT** está disponível somente no nível de mapeamento 2.1 em diante. O *value* de **INLINE-MAXOCCURS-LIMIT** pode ser um número inteiro positivo no intervalo de 0 a 32.767. Um valor 0 indica que o mapeamento sequencial não é usado. Um valor 1 garante que elementos opcionais sejam mapeados de forma sequencial. Se o *value* do atributo maxItems for maior do que o *value* de **INLINE-MAXOCCURS-LIMIT**, o mapeamento baseado em contêiner será usado; caso contrário, o mapeamento sequencial será usado.

Se você decidir que deseja que listas de repetições variáveis sejam mapeadas de forma sequencial, considere o comprimento de um único item de dados recorrentes. Se você tiver poucas instâncias de comprimento longo, o mapeamento baseado em contêiner será preferível; se você tiver muitas instâncias de comprimento curto, o mapeamento sequencial provavelmente será preferível.

JSONTRANSFRM = *value*

Este parâmetro é obrigatório para o modo vinculável (LINK), mas inválido para os modos de solicitação-resposta e RESTful. Ele indica o nome que é usado para o recurso de pacote configurável **JSONTRANSFRM** no CICS.

JSON-SCHEMA = *value*

O nome completo do arquivo z/OS UNIX a partir do qual o esquema JSON é lido. Se ele ainda não existir, DFHJS2LS criará o arquivo, mas não a estrutura de diretório.

LANG = **COBOL**

Especifica que a linguagem de programação da estrutura de linguagem de alto nível é COBOL.

LANG = **PLI-ENTERPRISE**

Especifica que a linguagem de programação da estrutura de linguagem de alto nível é Enterprise PL/I.

LANG = **PLI-OTHER**

Especifica que a linguagem de programação da estrutura de linguagem de alto nível é um nível de PL/I diferente de Enterprise PL/I.

LANG = **C**

Especifica que a linguagem de programação da estrutura de linguagem de alto nível é C.

LANG = **CPP**

Especifica que a linguagem de programação da estrutura de linguagem de alto nível é C++.

LOGFILE = *value*

O nome completo do z/OS UNIX do arquivo no qual DFHJS2LS grava seu log de atividades e suas informações de rastreamento. Se ele ainda não existir, DFHJS2LS criará o arquivo, mas não a estrutura de diretório.

Geralmente, você não usa esse arquivo, mas ele poderá ser solicitado pela organização de serviço da IBM caso você encontre problemas com o DFHJS2LS.

MAPPING-LEVEL = { **1.0** | **1.1** | **1.2** | **2.0** | **2.1** | **2.2** | **3.0** | **4.0** | **4.1** }

Especifica o nível de mapeamento para o assistente a ser usado ao gerar a ligação JSON e as estruturas de linguagem. Você deve usar o nível de mapeamento mais recente disponível; para DFHJS2LS, você deve usar um nível de mapeamento 3.0 ou superior.

3.0 Este é o nível de mapeamento mínimo que você pode usar com DFHJS2LS.

4.0 Use este nível de mapeamento com uma região do CICS TS 5.2 quando você desejar usar UTF-16.

4.1 Para suporte de matriz truncável, use este nível de mapeamento com uma região CICS TS 5.2 que tenha o APAR PI67641 aplicado ou qualquer região CICS TS 5.3 ou posterior.

MAPPING-OVERRIDES = { SAME-AS-MAPPING-LEVEL | HYPHENS-AS-UNDERScores | INTEGER-AS-PIC9 | LESS-DUP-NAMES | UNDERScores-AS-HYPHENS }

Especifica se o comportamento padrão é substituído para o nível de mapeamento especificado ao gerar estruturas de linguagem.

SAME-AS-MAPPING-LEVEL

Este parâmetro gera estruturas de linguagem no mesmo estilo que o nível de mapeamento. Este é o padrão.

HYPHENS-AS-UNDERScores

Para PL/I somente. Este parâmetro converte quaisquer hifens no esquema JSON em sublinhados em vez do caractere X, para melhorar a capacidade de leitura das estruturas de linguagem PL/I geradas. Para obter mais informações, consulte Esquema JSON para mapeamento de PL/I.

INTEGER-AS-PIC9

Apenas para COBOL e DFHJS2LS. Este parâmetro gera estruturas de linguagem que contêm valores de número inteiro do esquema JSON como numerais em vez de caracteres alfanuméricos.

LESS-DUP-NAMES

Este parâmetro gera nomes de campo de estrutura não estruturais com `_value` no final do nome para ativar a referência direta ao campo. Por exemplo, na estrutura de linguagem PLI a seguir, quando **MAPPING-OVERRIDES=LESS-DUP-NAMES** é especificado, o campo de nível 12 `streetName` é sufixado com `_value` :

```
09 streetName,  
12 streetName CHAR(255) VARYING  
UNALIGNED,  
12 filler BIT (7),  
12 attr_nil_streetName_value BIT (1),
```

A estrutura resultante é conforme a seguir:

```
09 streetName,  
12 streetName_value CHAR(255) VARYING  
UNALIGNED,  
12 filler BIT (7),  
12 attr_nil_streetName_value BIT (1),
```

UNDERScores-AS-HYPHENS

Esta opção é ativada automaticamente no nível de mapeamento 4.0.

Apenas para COBOL. Este parâmetro converte quaisquer sublinhados no esquema JSON em hifens, em vez do caractere X, para melhorar a capacidade de leitura das estruturas de linguagem COBOL geradas. Se ocorrerem quaisquer conflitos de nomes de campo, os campos serão numerados para garantir que sejam exclusivos. Para obter mais informações, consulte Esquema JSON para mapeamento COBOL.

MINIMUM-RUNTIME-LEVEL = { MINIMUM | 3.0 | 4.0 | 4.1 | CURRENT }

Especifica o ambiente de tempo de execução mínimo do CICS no qual a ligação JSON pode ser implementada. Se você selecionar um nível que não corresponde ao outros parâmetros especificados, receberá uma mensagem de erro. As opções que podem ser selecionadas são as seguintes:

MINIMUM

O menor nível de tempo de execução possível do CICS é alocado automaticamente com base nos parâmetros que você especificou.

- 3,0 Especifique o nível de tempo de execução 3.0 ou superior se deseja usar o assistente CICS JSON e tirar vantagem de mapeamentos de dados avançados.
- 4.0 O arquivo de ligação de serviço da web gerado é implementado com sucesso em um CICS TS 5.2 ou em uma região posterior. Com este nível de tempo de execução, é possível usar um nível de mapeamento 4.0 ou anterior para o parâmetro **MAPPING-LEVEL**. É possível utilizar qualquer parâmetro opcional neste nível.
- 4.1 O arquivo de ligação de serviço da web gerado é implementado com sucesso em uma região do CICS TS 5.2 que tenha o APAR PI67641 aplicado ou em qualquer região do CICS TS 5.3 ou posterior. Com este nível de tempo de execução, é possível utilizar um nível de mapeamento 4.1 ou anterior para o parâmetro **MAPPING-LEVEL**.

CURRENT

Use este nível de tempo de execução para implementar a ligação JSON gerada em uma região CICS que possui o mesmo ambiente de tempo de execução que a região usada para gerar a ligação JSON.

OVERWRITE-OUTPUT = { NO | YES }

Controla se os **BUNDLES** existentes do CICS no sistema de arquivos podem ser sobrescritos.

NO Qualquer **BUNDLE** existente não é substituído. Se um **BUNDLE** existente for localizado, o DFHJS2LS emitirá a mensagem de erro DFHPI9689E e finalizará.

YES Qualquer **BUNDLE** existente é substituído. Se um **BUNDLE** existente for localizado, a mensagem DFHPI9683W será emitida para informá-lo que o arquivo foi substituído.

PDSCP = *value*

Especifica a página de códigos que é usada nos membros do conjunto de dados particionados, em que *value* é um número de CCSID ou um número da página de códigos Java. Se você não especificar esse parâmetro, a página de códigos do z/OS UNIX System Services será usada. Por exemplo, você pode especificar PDSCP=037.

PDSLIB = *value*

Especifica o nome do conjunto de dados particionados que contém a linguagem de alto nível gerada.

PDSMEM = *value*

Especifica o prefixo de 1 a 6 caracteres que o DFHJS2LS usa para gerar o nome do membro do conjunto de dados particionados que conterá as estruturas de linguagem de alto nível.

O DFHJS2LS gera um membro do conjunto de dados particionados para cada operação. Ela gera o nome do membro anexando um número de dois dígitos no prefixo.

STRUCTURE = { PDSMEM_value | data }

O nome da estrutura de dados de nível superior em C e C++. O valor padrão é o valor do parâmetro **PDSMEM**.

WIDE-COMP3 = { FULL | NO | YES }

Controla o tamanho máximo do comprimento variável decimal compactado na estrutura de linguagem COBOL ou PL/I gerada.

FULL Para COBOL e PL/I. DFHJS2LS gera um campo decimal compactado

que é grande o suficiente para conter todos os valores válidos. O tamanho máximo são 31 dígitos. Este é o padrão.

NO Apenas para COBOL. DFHJS2LS limita o comprimento variável decimal compactado a 18 ao gerar a estrutura de linguagem COBOL tipo COMP-3. Se o tamanho decimal compactado for maior que 18, a mensagem DFHPI9022W será emitida para indicar que o tipo especificado está sendo limitado a um total de 18 dígitos.

YES Apenas para COBOL. DFHJS2LS suporta o tamanho máximo de 31 ao gerar a estrutura de linguagem COBOL tipo COMP-3.

Nota: As opções NO e YES geram campos que são incapazes de representar todos os valores válidos; a opção FULL evita esse problema. No entanto, a opção FULL permite que alguns valores inválidos sejam representados no campo decimal compactado. Por exemplo, se um esquema indicar que há um máximo de cinco dígitos e um máximo de dois dígitos fracionários, a opção FULL gerará um campo decimal compactado que permite sete dígitos e isso permite espaço para valores válidos, como 25000 e 999,99, mas também fornece espaço para alguns valores inválidos como 9999,99. Quando você usar a opção FULL, tome cuidado para não gerar valores inválidos em dados do aplicativo.

Exemplo

```
//JS2LS JOB '  
accounting information  
'  
,  
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  
JSONTRANSFRM=EXAMPLE  
BUNDLE=/u/exampleapp/bundles/exampleBundle  
/*
```

Níveis de mapeamento para o assistente CICS JSON

Um mapeamento é o conjunto de regras que especifica como as informações são convertidas entre as estruturas de linguagem e esquemas JSON. Para se beneficiar dos mapeamentos mais sofisticados disponíveis, é recomendável configurar o parâmetro **MAPPING-LEVEL** no assistente CICS para o nível mais recente.

Cada nível de mapeamento herda a função do mapeamento anterior, com o nível mais alto de mapeamento oferecendo os melhores recursos disponíveis. O nível de mapeamento mais alto fornece mais controle sobre a conversão de dados no tempo de execução e remove restrições no suporte para determinados tipos de dados e propriedades JSON

É possível configurar o parâmetro **MAPPING-LEVEL** para um nível anterior se você deseja reimplementar aplicativos que foram ativados anteriormente nesse nível.

Limitações em Todos os Níveis de Mapeamento

- Os tipos de dados usados no esquema JSON devem ser declarados explicitamente.
- Referências de objeto JSON a documentos externos usando \$ref não são suportadas no esquema JSON.

Nível de mapeamento 4.1

O nível de mapeamento 4.1 é compatível com uma região do CICS TS 5.3 com o APAR PI67641 aplicado e superiores.

O nível de mapeamento 4.1 é incluído nos Assistentes de Serviços da Web, nos Assistentes XML e Assistentes JSON. Este nível de mapeamento implementa mapeamentos melhorados para matrizes simples gerados de modo ascendente a partir de copybooks existentes; ele também inclui a capacidade do CICS de detectar automaticamente armazenamento final não inicializado em matrizes e de omitir esses registros no formulário XML/JSON gerado.

DFHLS2WS, DFHWS2LS, DFHLS2SC, DFHSC2LS, DFHJS2LS e DFHLS2JS suportam os parâmetros **TRUNCATE=NULL-ARRAYS** e **TRUNCATE=NULL-ARRAY-VALUES**.

Se você especificar qualquer valor para **TRUNCATE=NULL-ARRAY-VALUES**, também deverá especificar **TRUNCATE=NULL-ARRAYS=ENABLED**.

Nível de mapeamento 4.0

Este nível de mapeamento fornece o suporte a seguir:

No nível de mapeamento 4.0 e superior, o DFHLS2JS suporta a cláusula COBOL OCCURS DEPENDING ON e suporta o mapeamento de matrizes de caracteres COBOL para esquemas JSON. É possível configurar este comportamento usando o parâmetro **CHAR-OCCURS** no assistente CICS JSON.

- Deve-se especificar o parâmetro **DATA-TRUNCATION=ENABLED**.
- OCCURS DEPENDING ON complexo não é suportado. Esta limitação significa que OCCURS DEPENDING ON é suportado somente para o último campo de uma estrutura.
- O CICS não suporta nomes qualificados (usando a palavra-chave 'OF') como o destino de uma cláusula OCCURS DEPENDING ON, por exemplo FIELD1 OF STRUCTURE1.
- O CICS não suporta a palavra-chave UNBOUNDED. Deve-se especificar um limite de número inteiro para o tamanho máximo da tabela esperado pelo aplicativo.

No nível de mapeamento 4.0 e superior, serviços da web JSON suportam a conversão de dados do aplicativo que são codificados usando UTF-16 Unicode.

- Quando você usa LS2JS, é possível ativar este comportamento usando tipos de dados específicos da linguagem para UTF-16.
- Quando você usa JS2LS, é possível ativar este comportamento configurando CCSID=1200.
- CICS suporta somente uma única página de códigos Unicode, “ UTF-16BE com IBM Private Use Area ” (CCSID 1200).
- A conversão de dados do aplicativo que são codificados usando UTF-8 não é suportada.

Nota: O DFHLS2JS não suporta a cláusula COBOL GROUP USAGE NATIONAL.

Nível de mapeamento 3.0 e superior

Este nível de mapeamento fornece o suporte a seguir:

- DFHJS2LS mapeia tipos de dados date-time para o formato CICS ASKTIME.
- O DFHLS2JS pode gerar um esquema JSON e uma ligação de serviço da web a partir de um aplicativo que usa muitos contêineres em vez de somente um contêiner.
- A tolerância de dados truncados é descrita por uma estrutura de dados de comprimento fixo. É possível configurar este comportamento usando o parâmetro **DATA-TRUNCATION** no assistente do CICS.

Nível de Mapeamento 2.2 e Superior

Essa opção é mantida para compatibilidade com serviços da web SOAP. Ela não é recomendada para uso com JSON.

Nível de Mapeamento 2.1 e Superior

Essa opção é mantida para compatibilidade com serviços da web SOAP. Ela não é recomendada para uso com JSON.

Este nível de mapeamento inclui maior controle sobre a maneira como o conteúdo variável é manipulado com o novo parâmetro **INLINE-MAXOCCURS-LIMIT** e novos valores no parâmetro **CHAR-VARYING**.

No nível de mapeamento 2.1 e superior, o DFHJS2LS oferece o seguinte suporte novo e aprimorado para matrizes:

- O parâmetro **INLINE-MAXOCCURS-LIMIT**
- A propriedade `minItems`

O parâmetro **INLINE-MAXOCCURS-LIMIT** especifica se listas de repetição variável são mapeadas sequencialmente. Para obter mais informações sobre o mapeamento de conteúdo de repetição variável sequencial, consulte “Matrizes variáveis de elementos no DFHJS2LS” na página 282.

No nível de mapeamento 2.1 e superior, o DFHLS2JS suporta os mapeamentos JSON a seguir:

- Campos FILLER em COBOL e PL/I são ignorados
- Um valor de COLLAPSE para o parâmetro **CHAR-VARYING**
- Um valor de BINARY para o parâmetro **CHAR-VARYING**

Campos FILLER em COBOL e PL/I são ignorados; eles não aparecem no esquema JSON gerado e um intervalo apropriado é deixado nas estruturas de dados no tempo de execução.

COLLAPSE faz com que o CICS ignore espaços à direita em campos de texto.

BINARY fornece suporte para campos binários. Esse valor é útil ao converter COBOL em um esquema JSON. Esta opção está disponível somente nas matrizes de caracteres SBCS e permite que a matriz seja mapeada para uma sequência JSON de comprimento fixo contendo dados codificados em Base64 em vez de para uma sequência normal.

Nível de Mapeamento 1.2 e Superior

Essa opção é mantida para compatibilidade com serviços da web SOAP. Ela não é recomendada para uso com JSON.

Maior controle está disponível na maneira como dados de caracteres e binários são transformados no tempo de execução com esses parâmetros adicionais nas ferramentas em lote:

- **CHAR-VARYING**
- **CHAR-VARYING-LIMIT**
- **CHAR-MULTIPLIER**
- **DEFAULT-CHAR-MAXLENGTH**

Se você decidir usar o parâmetro **CHAR-MULTIPLIER** em DFHJS2LS, observe se as regras a seguir se aplicam após o valor desse parâmetro ser usado para calcular a quantidade de espaço necessária para dados de caractere.

- O DFHJS2LS fornece esses mapeamentos:
 - Tipos de dados de caractere de comprimento variável que possuem um comprimento máximo de mais de 32767 bytes são mapeados para um contêiner. É possível usar o parâmetro **CHAR-VARYING-LIMIT** para configurar um limite inferior. Um campo de 16 bytes é criado na estrutura de linguagem para armazenar o nome do contêiner. No tempo de execução, os dados de caracteres são armazenados em um contêiner e o nome do contêiner é colocado na estrutura de linguagem.
 - Os tipos de dados de caractere de comprimento variável que possuem um comprimento máximo de menos de 32.768 bytes são mapeados para uma estrutura **VARYING** para todas as linguagens, exceto C/C++ e Enterprise PL/I. No C/C++, esses tipos de dados são mapeados para sequências com terminação nula e no Enterprise PL/I esses tipos de dados são mapeados para estruturas **VARYINGZ**. É possível usar o parâmetro **CHAR-VARYING** para selecionar o modo como os dados de caractere de comprimento variável são mapeados.
 - Os dados binários de comprimento variável que possuem um comprimento máximo de menos de 32768 bytes são mapeados para uma estrutura **VARYING** para todas as linguagens. Se o comprimento máximo for igual ou maior do que 32.768 bytes, os dados serão mapeados para um contêiner. Um campo de 16 bytes é criado na estrutura de linguagem para armazenar o nome do contêiner. No tempo de execução, os dados binários são armazenados em um contêiner e o nome do contêiner é colocado na estrutura de linguagem.

Se você tiver tipos de dados de caractere no esquema JSON que não possuem um comprimento associado a eles, poderá designar um comprimento padrão usando o parâmetro **DEFAULT-CHAR-MAXLENGTH** no DFHJS2LS.

O DFHLS2JS fornece esses mapeamentos:

- Campos de caracteres mapeiam para um tipo de dados string e podem ser processados como campos de comprimento fixo ou sequências com terminação nula no tempo de execução. É possível usar o parâmetro **CHAR-VARYING** para selecionar o modo como os dados de caractere de comprimento variável são manipulados no tempo de execução para todas as linguagens, exceto PL/I.
- Os tipos de dados Base64Binary mapeiam para um contêiner se o comprimento máximo dos dados é maior do que 32767 bytes ou quando o comprimento não

está definido. Se o comprimento dos dados é 32767 ou menos, o tipo de dados base64Binary é mapeado para uma estrutura VARYING para todas as linguagens.

Nível de Mapeamento 1.1 e Superior

Essa opção é mantida para compatibilidade com serviços da web SOAP. Ela não é recomendada para uso com JSON.

Este nível de mapeamento fornece mapeamento aprimorado de caractere JSON e tipos de dados binários, em específico, ao mapear dados de comprimento variável que possui as propriedades maxLength e minLength definidas com valores diferentes no esquema JSON. Os dados são manipulados da seguinte forma:

- Os tipos de dados de caractere e binários que possuem um comprimento fixo que é maior do que 16 MB mapeiam para um contêiner para todas as linguagens, exceto PL/I. No PL/I, tipos de dados de caractere e binários de comprimento fixo que são maiores do que 32767 bytes são mapeados para um contêiner. Um campo de 16 bytes é criado na estrutura de linguagem para armazenar o nome do contêiner. No tempo de execução, os dados de comprimento fixo são armazenados em um contêiner e o nome do contêiner é colocado na estrutura de linguagem.

Como os contêineres são variáveis no comprimento, os dados de comprimento fixo que são mapeados para um contêiner não são preenchidos com espaços ou nulos, ou truncados, para corresponder ao comprimento fixo especificado no esquema JSON. Se o comprimento dos dados for significativo, será possível gravar seu aplicativo para verificá-lo ou ativar a validação na região CICS. A validação tem um impacto significativo no desempenho.

Apenas Nível de Mapeamento 1.1

Essa opção é mantida para compatibilidade com serviços da web SOAP. Ela não é recomendada para uso com JSON.

Este nível de mapeamento fornece mapeamento aprimorado de caractere JSON e tipos de dados binários, em específico, ao mapear dados de comprimento variável que possui as propriedades maxLength e minLength definidas com valores diferentes no esquema JSON. Os dados são manipulados das maneiras a seguir:

- Tipos de dados binários de comprimento variável mapeiam para um contêiner. Um campo de 16 bytes é criado na estrutura de linguagem para armazenar o nome do contêiner. No tempo de execução, os dados binários são armazenados em um contêiner e o nome do contêiner é colocado na estrutura de linguagem.
- Os tipos de dados de caractere de comprimento variável que possuem um comprimento máximo maior que 32767 bytes mapeiam para um contêiner. Um campo de 16 bytes é criado na estrutura de linguagem para armazenar o nome do contêiner. No tempo de execução, os dados de caractere são armazenados em um contêiner e o nome do contêiner é colocado na estrutura de linguagem.
- Os tipos de dados de caractere e binários que possuem um comprimento fixo de menos de 16 MB mapeiam para campos de comprimento fixo para todas as linguagens, exceto PL/I. Em PL/I, os tipos de dados de caractere e binários de comprimento fixo que têm 32767 bytes ou menos mapeiam para campos de comprimento fixo.

- O CICS codifica e decodifica dados no formato `hexBinary`, mas não no formato `base64Binary`. Tipos de dados `Base64Binary` no mapa de esquema JSON para um campo na estrutura de linguagem. O tamanho do campo é calculado usando a fórmula: $4 \times (\text{ceil}(z/3))$ em que:

- z é o comprimento do tipo de dados no esquema JSON.
- $\text{ceil}(x)$ é o menor número inteiro maior ou igual a x .

Se o comprimento de z for maior do que 24566 bytes, a estrutura de linguagem resultante falhará ao compilar. Se você possui dados `base64Binary` que são maiores do que 24566 bytes, é recomendável usar um nível de mapeamento igual a 1.2. Com nível de mapeamento 1.2, é possível mapear os dados do `base64Binary` para um contêiner em vez de usar um campo na estrutura de linguagem.

Apenas Nível de Mapeamento 1.0

Essa opção é mantida para compatibilidade com serviços da web SOAP. Ela não é recomendada para uso com JSON.

Observe as limitações a seguir, que foram modificadas em níveis de mapeamento posteriores:

- O DFHJS2LS mapeia tipos de dados de caractere e binários no esquema JSON para campos de comprimento fixo na estrutura de linguagem. Examine este esquema JSON parcial:

```
"example":{
  "type": "string",
  "maxLength":33000
```

Esse esquema JSON parcial aparece em uma estrutura de linguagem COBOL como:

```
15 example PIC X(33000)
```

- O CICS codifica e decodifica dados no formato `hexBinary`, mas não no formato `base64Binary`. O DFHJS2LS mapeia dados `Base64Binary` para um campo de caracteres de comprimento fixo, cujo conteúdo deve ser codificado ou decodificado pelo programa de aplicativo.
- O DFHLS2JS interpreta campos de caracteres e binários na estrutura de linguagem como campos de comprimento fixo e mapeiam esses campos para sequências JSON que possuem uma propriedade `maxLength`. No tempo de execução, os campos na estrutura de linguagem são preenchidos com espaços ou nulos se dados insuficientes estão disponíveis.

Linguagem de alto nível e mapeamento de esquema JSON

Use o assistente CICS JSON para gerar mapeamentos entre as estruturas de linguagem de alto nível e os esquemas JSON. O assistente CICS JSON também gera esquemas JSON a partir de estruturas de dados de linguagem de alto nível ou vice-versa.

Os programas utilitários DFHJS2LS e DFHLS2JS são conhecidos coletivamente como o assistente CICS JSON.

- DFHLS2JS mapeia as estruturas de linguagem de alto nível para esquemas JSON.
- DFHJS2LS mapeia esquemas JSON para estruturas de linguagem de alto nível.

Os dois mapeamentos não são simétricos:

- Se você processar uma estrutura de dados de linguagem com DFHLS2JS e, em seguida, processar o esquema JSON resultante com o DFHJS2LS, não espere que a estrutura de dados final seja igual à original.
- Se você processar um esquema JSON com DFHJS2LS e, em seguida, processar a estrutura de linguagem resultante com o DFHLS2JS, não espere que o esquema JSON final seja igual ao original.
- Em alguns casos, DFHJS2LS gera estruturas de linguagem que não são suportadas pelo DFHLS2JS.

Deve-se codificar estruturas de linguagem de alto nível que sejam processadas pelo DFHLS2JS de acordo com as regras da linguagem, conforme implementado nos compiladores de linguagem que o CICS suporta.

Mapeamento de esquema COBOL para JSON:

O programa utilitário DFHLS2JS suporta mapeamentos entre as estruturas de dados COBOL e as definições de esquema JSON.

Os nomes COBOL são convertidos em nomes JSON de acordo com as regras a seguir:

- Nomes duplicados se tornam exclusivos pela adição de um ou mais dígitos numéricos.
Por exemplo, duas instâncias de year se tornam year e year1.
- Hifens são substituídos por sublinhados. Sequências de hifens contíguos são substituídas por sublinhados contíguos.
Por exemplo, current-user--id se torna current_user_id.
- Segmentos de nomes que são delimitados por hifens e que contêm somente caracteres em maiúsculas são convertidos em minúsculas.
Por exemplo, CA-REQUEST-ID se torna ca_request_id.
- Um sublinhado inicial é incluído em nomes que iniciam com um caractere numérico.
Por exemplo, 9A-REQUEST-ID se torna _9a_request_id.

O CICS mapeia elementos de descrição de dados COBOL para elementos de esquema de acordo com a tabela a seguir. Elementos de descrição de dados COBOL que não são mostrados na tabela não são suportados pelo DFHLS2JS. As seguintes restrições também se aplicam:

- Os itens de descrição de dados com os números de nível 66 e 77 não são suportados. Os itens de descrição de dados com um número de nível 88 são ignorados.
- As seguintes cláusulas nas entradas de descrição de dados não são suportadas:
REDEFINES
RENAMES; que é o nível 66
DATE FORMAT
- As seguintes cláusulas nos itens de descrição de dados são ignoradas:
BLANK WHEN ZERO
JUSTIFIED
VALUE
- A cláusula SIGN, SIGN TRAILING, é suportada. A cláusula SIGN, SIGN LEADING, é suportada somente quando o nível de mapeamento especificado em DFHLS2JS é 1.2 ou superior.

- SEPARATE CHARACTER é suportado em um nível de mapeamento de 1.2 ou superior para ambas as cláusulas SIGN TRAILING e SIGN LEADING.
- As seguintes frases na cláusula USO não são suportadas:
 - OBJECT REFERENCE
 - POINTER
 - FUNCTION-POINTER
 - PROCEDURE-POINTER
- As frases a seguir na cláusula USAGE são suportadas em um nível de mapeamento de 1.2 ou superior:
 - COMPUTATIONAL-1
 - COMPUTATIONAL-2
- Os únicos caracteres de PICTURE suportados para os itens de descrição de dados DISPLAY e COMPUTATIONAL-5 são 9, S e Z.
- Os caracteres de PICTURE que são suportados para os itens de descrição de dados PACKED-DECIMAL são 9, S, V e Z.
- Os únicos caracteres de PICTURE que são suportados para os itens de descrição de dados numéricos editados são 9 e Z.
- Se o parâmetro MAPPING-LEVEL estiver configurado como 1.2 ou superior e o parâmetro CHAR-VARYING estiver configurado como NULL, as matrizes de caracteres serão mapeadas para uma sequência e serão processadas como sequências com terminação nula.
- Se o parâmetro MAPPING-LEVEL estiver configurado como 1.2 ou superior e o parâmetro CHAR-VARYING estiver configurado como BINARY, as matrizes de caracteres serão mapeadas para uma sequência e serão processadas como dados binários.
- Se o parâmetro MAPPING-LEVEL estiver configurado como 1.2 ou superior e o parâmetro CHAR-VARYING estiver configurado como COLLAPSE, o espaço em branco final será ignorado para as sequências.
- A cláusula OCCURS DEPENDING ON é suportada em um nível de mapeamento de 4.0 ou superior. OCCURS DEPENDING ON complexo não é suportado. Isso significa que OCCURS DEPENDING ON é suportado somente para o último campo de uma estrutura.
- A cláusula OCCURS INDEXED BY é suportada em qualquer nível de mapeamento.

Descrição de Dados COBOL	Definição 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>)	

Descrição de Dados COBOL	Definição de esquema JSON
PIC S9 DISPLAY PIC S99 DISPLAY PIC S999 DISPLAY PIC S9999 DISPLAY PIC S9(n) DISPLAY PIC S9(n) COMP PIC S9(n) COMP-4 PIC S9(n) COMP-5 PIC S9(n) BINARY	<pre>"type": "integer", "minimum":- (n + 1), "maximum": n</pre> <p>em que n é o valor máximo que pode ser representado pelo padrão de '9' caracteres.</p>
PIC 9 DISPLAY PIC 99 DISPLAY PIC 999 DISPLAY PIC 9999 DISPLAY PIC 9(n) DISPLAY PIC 9(n) COMP PIC 9(n) COMP-4 PIC 9(n) COMP-5 PIC 9(n) BINARY	<pre>"type": "integer", "minimum":0, "maximum": n</pre> <p>em que n é o valor máximo que pode ser representado pelo padrão de '9' caracteres.</p>

Descrição de Dados COBOL	Definição 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>em que:</p> <p><i>x</i> é o valor mínimo que pode ser representado pelo padrão de '9' caracteres</p> <p><i>y</i> é o valor máximo que pode ser representado pelo padrão de '9' caracteres</p> <p><i>z</i> é a menor unidade disponível = 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>em que:</p> <p><i>y</i> é o valor máximo que pode ser representado pelo padrão de '9' caracteres</p> <p><i>z</i> é a menor unidade disponível = 1/10 "</p>
PIC S9(15) COMP-3 Suportado no nível de mapeamento 3.0 quando DATETIME=PACKED15	<pre>"type": "string", "format": "date-time"</pre> <p>O formato do registro de data e hora é definido pelo RFC3339.</p>
PIC S9(<i>m</i>)V9(<i>n</i>) DISPLAY Suportado no nível de mapeamento 1.2 e superior	<pre>"type": "number", "format": "decimal", "minimum": x , "maximum": y , "multipleOf": z</pre> <p>em que:</p> <p><i>x</i> é o valor mínimo que pode ser representado pelo padrão de '9' caracteres</p> <p><i>y</i> é o valor máximo que pode ser representado pelo padrão de '9' caracteres</p> <p><i>z</i> é a menor unidade disponível = 1/10 "</p>

Descrição de Dados COBOL	Definição de esquema JSON
<p>COMP-1</p> <p>Suportado no nível de mapeamento 1.2 e superior</p> <p>Nota: A representação de dados do IBM Hexadecimal Floating Point (HFP) não é exatamente igual à representação IEEE-754-1985 usada para JSON. Alguns valores podem não ser convertidos exatamente de uma representação para a outra.</p> <p>Alguns valores extremamente grandes ou pequenos podem não ser válidos para tipos de dados flutuantes. Alguns valores podem perder precisão quando convertidos para ou a partir de uma representação HFP. Se conversões precisas forem importantes, considere substituir o uso de tipos de dados COMP-1 por alternativas de precisão fixas.</p>	<pre>"type": "number", "format": "float"</pre>
<p>COMP-2</p> <p>Suportado no nível de mapeamento 1.2 e superior</p> <p>Nota: A representação de dados do IBM Hexadecimal Floating Point (HFP) não é exatamente igual à representação IEEE-754-1985 usada para JSON. Alguns valores podem não ser convertidos exatamente de uma representação para a outra.</p> <p>Alguns valores extremamente grandes ou pequenos podem não ser válidos para tipos de dados duplos. Alguns valores podem perder precisão quando convertidos para ou a partir de uma representação HFP. Se conversões precisas forem importantes, considere substituir o uso de tipos de dados COMP-2 por alternativas de precisão fixas .</p>	<pre>"type": "number", "format": "double"</pre>

Descrição de Dados COBOL	Definição de esquema JSON
<i>data description</i> OCCURS <i>n</i> TIMES	<p>No nível de mapeamento 4.0 e inferior</p> <p>Para primitivas:</p> <pre> "type": "array" "maxItems": n "minItems": n "items": { "tipo": "objeto", "properties": { name : { data description JSON } } "required": [name] } </pre> <p>Para estruturas:</p> <pre> "type": "array" "maxItems": n "minItems": n "items": { data description JSON } </pre> <p>Em que <i>data description JSON</i> é a representação de esquema JSON do COBOL <i>data description</i> e <i>name</i> é o nome do COBOL <i>data description</i>.</p> <p>No nível de mapeamento 4.1 e superior</p> <p>TRUNCATE-NULL-ARRAYS = DISABLED</p> <p>As matrizes estruturadas e primitivas são mapeadas conforme a seguir.</p> <pre> "type": "array" "maxItems": n "minItems": n "items": { data description JSON } </pre> <p>TRUNCATE-NULL-ARRAYS = ENABLED</p> <p>As matrizes primitivas são mapeadas conforme acima e as matrizes estruturadas conforme a seguir.</p> <pre> "type": "array" "maxItems": n "minItems": 0 "items": { data description JSON } </pre>

Descrição de Dados COBOL	Definição de esquema JSON
<i>data description</i> OCCURS <i>n</i> TO <i>m</i> TIMES DEPENDING ON <i>t</i> Suportado no nível de mapeamento 4.0	<pre>"field-name" : { "type": "array" , "maxItems": <i>m</i> "minItems": <i>n</i> "items": { ... } }</pre> O conteúdo do item de matriz depende do tipo de dados usado.
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	Quando CHAR-OCCURS =STRING: <pre>"field-name" : { "type": "string" , "maxLength": <i>n</i> }</pre> Ele é uma sequência.

Descrição de Dados COBOL	Definição de esquema JSON
	<p>Quando CHAR-OCCURS =ARRAY:</p> <p>No nível de mapeamento 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>Isto é uma matriz de caracteres únicos.</p> <p>No nível de mapeamento 4.1 e superior</p> <p>TRUNCATE-NULL-ARRAYS = DISABLED</p> <p>As matrizes estruturadas e primitivas são mapeadas conforme a seguir.</p> <pre> "type":"array" "maxItems": n "minItems": n "items":{ data description JSON } </pre> <p>TRUNCATE-NULL-ARRAYS = ENABLED</p> <p>As matrizes primitivas são mapeadas conforme acima e as matrizes estruturadas conforme a seguir.</p> <pre> "type":"array" "maxItems": n "minItems": 0 "items":{ data description JSON } </pre>

Descrição de Dados COBOL	Definição 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>	Quando CHAR-OCCURS =STRING: <i>"field-name"</i> :{ "type": "string" , "maxLength": <i>m</i> "minLength": <i>n</i> }
PIC N(<i>n</i>) USAGE NATIONAL Quando CHAR-USAGE =NATIONAL : PIC N(<i>n</i>)	No nível de mapeamento 4.0 e superior: "type": "string", "maxLength": <i>n</i> No tempo de execução, CICS preenche o campo de estrutura de dados do aplicativo com dados UTF-16.

Esquema JSON para mapeamento COBOL:

O programa utilitário DFHJS2LS suporta mapeamentos entre o esquema JSON e as estruturas de dados COBOL.

Os assistentes do CICS geram nomes de campo válidos e exclusivos para variáveis COBOL a partir dos nomes de elemento de esquema usando as regras a seguir:

1. Palavras reservadas COBOL são prefixadas com ' X' inicial.
Por exemplo, DISPLAY se torna XDISPLAY.
2. Caracteres diferentes de A-Z, a-z, 0-9 ou hífen são substituídos por ' X' inicial.
Por exemplo, monthly_total se torna monthlyXtotal.
3. Se o último caractere for um hífen, ele será substituído por ' X' inicial.
Por exemplo, ca-request- se torna ca-requestX.

4. Nomes duplicados no mesmo escopo se tornam exclusivos pela adição de um ou dois dígitos numéricos na segunda instância e nas instâncias subsequentes do nome.
Por exemplo, três instâncias de year se tornam year , year1 e year2.
5. Um esquema JSON especifica que uma variável possui variação de cardinalidade se possui um valor de "type" igual a "array" e as palavras-chave "minItems" e "maxItems" foram omitidas ou possuem valores diferentes. Se o esquema especifica que a variável possui variação de cardinalidade, os nomes de campo são criados com sufixos "_cont" e "_num".
Para obter mais informações, consulte “Matrizes variáveis de elementos no DFHJS2LS” na página 282.
6. Um esquema JSON especifica que uma variável é opcional se ela não aparece na matriz de palavra-chave "required" que está associada ao tipo de "object" de esquema JSON de inclusão. Para campos opcionais, um campo adicional é gerado com um sufixo _num incluído no nome de elemento. No tempo de execução, ele é zero para indicar que o valor estava ausente nos dados JSON e não zero se o valor estava presente nos dados JSON.
7. Os nomes de campo são limitados a 28 caracteres. Se um nome gerado, incluindo o prefixo e o sufixo, exceder esse comprimento, o nome de elemento será truncado.

DFHJS2LS mapeia tipos de esquema para elementos de descrição de dados COBOL usando o nível de mapeamento especificado de acordo com a tabela a seguir. Observe os pontos a seguir:

- Se o parâmetro **MAPPING-LEVEL** for configurado como 1.2 ou superior e o parâmetro **CHAR-VARYING** for configurado como NULL, os dados de caractere de comprimento variável serão mapeados para sequências com terminação nula e um caractere extra será alocado para o terminador NULL.
- Se o parâmetro **MAPPING-LEVEL** for configurado como 1.2 ou superior e o parâmetro **CHAR-VARYING** for configurado como YES, os dados de caractere de comprimento variável serão mapeados para dois elementos relacionados: um campo de comprimento e um campo de dados. Por exemplo:

```
"textString": {
  "type": "string",
  "maxLength": 10000,
  "minLength": 1
}
```

mapeia para:

```
15 textString-length PIC S9999 COMP-5 SYNC
15 textString PIC X(10000)
```

Palavra-chave do Esquema JSON	Descrição de Dados COBOL
Todos de: "type": "null" "type": [] "enum": [] "allOf" "anyOf" "noneOf" "not"	Não-suportado

Palavra-chave do Esquema JSON	Descrição de Dados COBOL
"\$schema": "http://json-schema.org/draft-04/schema#"	Esta palavra-chave é ignorada, mas assume-se que ela é compatível com a especificação de Esquema JSON 04 de rascunho.
"title": "same text" "description": "more text"	Essas palavras-chave são ignoradas.
"format": "<predefined values>"	A palavra-chave "format" é utilizada para modificar a estrutura gerada ou o valor de tempo de execução. Consulte as informações posteriormente nesta tabela para o uso suportado de "format".
"type": "array", "items": {<JSON Sub-schema>}, "additionalItems": false, "maxItems": m , "minItems": n	<p>A única forma de matriz JSON suportada atualmente é um número repetido dos mesmos valores de tipo. O <JSON Sub-schema> deve definir um "type" suportado, mas esse "type" não pode ser "array". Esta é uma restrição na estrutura de linguagem gerada.</p> <p>"additionalItems" é assumido como false e nenhum outro valor é suportado.</p> <p>Se "minItems" e "maxItems" estiverem presentes e forem iguais, a matriz será tratada como cardinalidade fixa, caso contrário, será tratada como cardinalidade variável. Consulte "Matrizes variáveis de elementos no DFHJS2LS" na página 282.</p>
"type": "array", "uniqueItems": true	"uniqueItems" não é suportado com matrizes JSON.
"type": "object", "additionalProperties": false, "properties": { ["<element name>": {<JSON Sub-schema> } [,]]* } "required": [["<element name> [,]]*]	<p>O único formato do objeto JSON que é suportado atualmente é um conjunto fixo de elementos nomeados.</p> <p>Isto gera uma estrutura (ou subestrutura) que usa os nomes de elemento.</p> <p>"additionalProperties" é assumido como false e nenhum outro valor é suportado.</p> <p>Qualquer elemento no objeto "properties" é considerado "optional" se ele não está na matriz "required" ou se nenhuma matriz "required" existe. Um elemento "optional" recebe uma cardinalidade variável de zero a X; em que X é 1 ou o número máximo de itens na matriz no qual esse item é definido como uma matriz. Consulte "Matrizes variáveis de elementos no DFHJS2LS" na página 282.</p>

Palavra-chave do Esquema JSON	Descrição de Dados COBOL
<pre>"type": "object", "maxProperties": m, "minProperties": n, "patternProperties": {}, "dependencies":</pre>	Nenhuma dessas palavras-chave é suportada com objetos JSON.
<pre>"type": "string" "maxLength": m "pattern": "regular expression>", "minLength": l</pre>	<p>PIC X(z)</p> <p>em que o valor de <i>z</i> é baseado em <i>m</i>, porém é dependente das configurações do parâmetro <i>CHAR-VARYING</i>.</p> <p><i>m</i> é baseado na palavra-chave "maxLength" e tratado como uma sequência de comprimento fixo.</p> <p>As restrições "pattern" e "minLength" são transmitidas para a estrutura de linguagem somente como um comentário.</p>
<pre>"type": "string" "maxLength": m</pre>	<p>Quando CCSID=1200 no nível de mapeamento 4.0 e superior:</p> <p>PIC N(z) USAGE NATIONAL</p> <p>em que o valor de <i>z</i> é baseado em <i>m</i>, porém é dependente das configurações do parâmetro CHAR-VARYING.</p> <p><i>m</i> é baseado na palavra-chave "maxLength" e tratado como uma sequência de comprimento fixo.</p>
<pre>"*name*":{ "type": "string", "format": "date-time" }</pre>	<p>PIC S9(15) COMP-3</p> <p>Todos suportados quando DATETIME=PACKED15</p> <p>Observe que "maxLength" e "minLength" não são suportados para este formato</p>
<pre>"*name*":{ "type": "string", "format": "uri" }</pre>	<p>PIC X(m)</p> <p>em que <i>m</i> é baseado na palavra-chave "maxLength" e tratado como uma sequência de comprimento fixo.</p> <p>Quando CCSID=1200 no nível de mapeamento 4.0 e superior:</p> <p>PIC N(m) USAGE NATIONAL</p> <p>em que <i>m</i> é baseado na palavra-chave "maxLength" e tratado como uma sequência de comprimento fixo.</p>
<pre>"*name*":{ "type": "string", "format": "base64Binary" }</pre>	<p>PIC X(m)</p> <p>em que <i>m</i> é baseado na palavra-chave "maxLength"</p>

Palavra-chave do Esquema JSON	Descrição de Dados COBOL
<pre>"*name*":{ "type": "string", "format":"hexBinary" }</pre>	<p>PIC X(<i>m</i>)</p> <p>em que <i>m</i> é baseado na palavra-chave "maxLength"</p>
<pre>"*name*":{ "type": "string", "format":"<predefined>" }</pre>	<p>PIC X(<i>m</i>)</p> <p>em que <i>m</i> é baseado na palavra-chave "maxLength" e tratado como sequência de comprimento fixo e em que <predefined> é um de: <i>e-mail</i>, <i>hostname</i>, <i>ipv4</i> ou <i>ipv6</i>. Um "padrão" relevante é usado e transmitido ao comentário.</p> <p>Quando CCSID=1200 no nível de mapeamento 4.0 e superior:</p> <p>PIC N(<i>m</i>) USAGE NATIONAL</p> <p>em que <i>m</i> é baseado na palavra-chave "maxLength" e tratado como sequência de comprimento fixo e em que <predefined> é um de: <i>e-mail</i>, <i>hostname</i>, <i>ipv4</i> ou <i>ipv6</i>. Um "padrão" relevante é usado e transmitido ao comentário.</p>
<pre>"type": "boolean"</pre>	<p>PIC X DISPLAY</p> <p>O valor x'00' significa false, x'01' significa true.</p>
<pre>"type": "integer", "maxExclusive": true, "minExclusive": true, "multipleOf": n</pre>	<p>As restrições "maxExclusive" e "minExclusive" são transmitidas à estrutura de linguagem somente como um comentário.</p> <p>"multipleOf" é ignorado.</p>
<pre>"type"="integer", minimum=0, maximum=255</pre>	<p>Nível de mapeamento 3.0 e abaixo:</p> <p>PIC X DISPLAY</p> <p>Nível de mapeamento 4.0 e acima (ou quando o parâmetro INTEGER-AS-PIC9 tiver sido especificado, independentemente do nível de mapeamento):</p> <p>PIC 9(<i>z</i>) COMP-5 SYNC</p> <p>ou</p> <p>PIC 9(<i>z</i>) DISPLAY</p> <p>em que $10^{(z-1)} < m \leq 10^z$</p>

Palavra-chave do Esquema JSON	Descrição de Dados COBOL
"type": "integer", minimum:-128, maximum:127	Nível de mapeamento 3.0 e abaixo: PIC X DISPLAY Nível de mapeamento 4.0 e acima (ou quando o parâmetro INTEGER-AS-PIC9 tiver sido especificado, independentemente do nível de mapeamento): PIC S9(z) COMP-5 SYNC ou PIC S9(z) DISPLAY em que $10^{(z-1)} < m \leq 10^z$
"type": "integer", minimum:0, maximum; m	PIC 9(z) COMP-5 SYNC ou PIC 9(z) DISPLAY em que $10^{(z-1)} < m \leq 10^z$
"type": "integer", mínimo:- m , máximo: m -1	PIC S9(z) COMP-5 SYNC ou PIC S9(z) DISPLAY em que $10^{(z-1)} < m \leq 10^z$
"type": "number", "maximum": m, "minimum": n, "maxExclusive": true, "minExclusive": true, "multipleOf": n	As restrições "maximum" , "minimum" , "maxExclusive" e "minExclusive" são transmitidas à estrutura de linguagem somente como um comentário. "multipleOf" é ignorado.
"type": "number" "format": "decimal"	PIC 9(p)V9(n) COMP-3 em que p e n são valores padrão.

Palavra-chave do Esquema JSON	Descrição de Dados COBOL
<pre>"type": "number" "format": "float"</pre>	<p>Nível de mapeamento 1.1 e inferior:</p> <ul style="list-style-type: none"> PIC X(32) <p>Nível de mapeamento 1.2 e posterior:</p> <ul style="list-style-type: none"> COMP-1 <p>Nota: A representação de dados do IBM Hexadecimal Floating Point (HFP) não é exatamente igual à representação IEEE-754-1985 usada para JSON. Alguns valores podem não ser convertidos exatamente de uma representação para a outra.</p> <p>Alguns valores extremamente grandes ou pequenos podem não ser válidos para tipos de dados flutuantes. Alguns valores podem perder precisão quando convertidos para ou a partir de uma representação HFP. Se conversões precisas forem importantes, considere substituir o uso de tipos de dados COMP-1 por alternativas de precisão fixas.</p>
<pre>"type": "number" "format": "double"</pre>	<p>Nível de mapeamento 1.1 e inferior:</p> <ul style="list-style-type: none"> PIC X(32) <p>Nível de mapeamento 1.2 e posterior:</p> <ul style="list-style-type: none"> COMP-2 <p>Nota: A representação de dados do IBM Hexadecimal Floating Point (HFP) não é exatamente igual à representação IEEE-754-1985 usada para JSON. Alguns valores podem não ser convertidos exatamente de uma representação para a outra.</p> <p>Alguns valores extremamente grandes ou pequenos podem não ser válidos para tipos de dados duplos. Alguns valores podem perder precisão quando convertidos para ou a partir de uma representação HFP. Se conversões precisas forem importantes, considere substituir o uso de tipos de dados COMP-2 por alternativas de precisão fixas .</p>

Nota: O CICS não pode transformar valores de número inteiro maiores do que o valor máximo para um longo sinalizado ($2^{63} - 1$), a menos que eles sejam colocados entre aspas.

Nota: Os valores mínimo e máximo especificados no esquema para tipos numéricos são utilizados somente para mapear para um tipo de dados COBOL. Os dados não são validados com relação a esses valores no tempo de execução.

Alguns dos tipos de esquema que são mostrados na tabela mapeiam para um formato COBOL de COMP-5 SYNC ou de DISPLAY, dependendo dos valores (se houver) que são especificados nas palavras-chave `minimum` e `maximum`:

- Para tipos sinalizados (`short`, `int`, e `long`), `DISPLAY` é usado quando o seguinte é especificado:

```
"maximum":  
  a  
"minimum":  
  -a
```

em que *a* é uma sequência de '9'.

- Para tipos não sinalizados (`unsignedShort`, `unsignedInt` e `unsignedLong`), `DISPLAY` é usado quando o seguinte é especificado:

```
"maximum":  
  a  
"minimum":0
```

em que *a* é uma sequência de '9'.

Quando qualquer outro valor é especificado, ou nenhum valor é especificado, `COMP-5 SYNC` é usado.

C e C++ para mapeamento de esquema JSON:

O programa utilitário `DFHLS2JS` suporta mapeamentos entre tipos de dados C e C++ e definições de esquema JSON.

Os nomes C e C++ são convertidos em nomes JSON de acordo com as seguintes regras:

1. Os caracteres que não são válidos em nomes de propriedades JSON são substituídos por 'X'.
Por exemplo, `monthly-total` se torna `monthlyXtotal`.
2. Nomes duplicados se tornam exclusivos pela adição de um ou mais dígitos numéricos.
Por exemplo, duas instâncias de `year` se tornam `year` e `year1`.

O `DFHLS2JS` mapeia tipos de dados C e C++ para os elementos de esquema de acordo com a tabela a seguir. Os tipos C e C++ que não são mostrados na tabela não são suportados pelo `DFHLS2JS`. O qualificador `_Packed` é suportado para estruturas. Estas restrições se aplicam:

- Os arquivos de cabeçalho devem conter uma instância `struct` de nível superior.
- Não é possível declarar um tipo de estrutura que contenha ele mesmo como um membro.
- Os seguintes tipos de dados C e C++ não são suportados:
 - `decimal`
 - `long double`
 - `wchar_t` (somente C++)
- Os seguintes são ignorados se estão presentes no arquivo de cabeçalho.

Especificadores de classe de armazenamento:

`auto`

register
static
extern
mutable

Qualificadores

const
volatile
_Export (somente C++)

Especificações de funções

inline (somente C++)
virtual (somente C++)

Valores iniciais

- O arquivo de cabeçalho não deve conter estes itens:
 - Uniãos
 - Declarações de classe
 - Tipos de dados de enumeração
 - Variáveis de tipo de ponteiro
 - Declarações de modelo
 - Macros predefinidas; isto é, macros com nomes que iniciam e terminam com dois caracteres de sublinhado (__)
 - A sequência de continuação de linha (um símbolo \ que é imediatamente seguido por um caractere de nova linha)
 - Declaradores de função de protótipo
 - Diretivas de pré-processadores
 - Campos de bits
 - A palavra-chave __cdecl (ou _cdecl) (somente C++)
- O programador do aplicativo deve utilizar um compilador de 32 bits para assegurar que um int seja mapeado para 4 bytes.
- As palavras-chave reservadas por C++ a seguir não são suportadas:
 - explicit
 - using
 - namespace
 - typename
 - typeid
- Se o parâmetro **MAPPING-LEVEL** estiver configurado como 1.2 ou superior e o parâmetro **CHAR-VARYING** estiver configurado como NULL, as matrizes de caracteres serão mapeadas para um string e serão processadas como sequências com terminação nula.
- Se o parâmetro **MAPPING-LEVEL** for configurado como 1.2 ou superior e o parâmetro **CHAR-VARYING** for configurado como BINARY, as matrizes de caracteres serão mapeadas para xsd:base64Binary e serão processadas como dados binários.
- Se o parâmetro **MAPPING-LEVEL** for configurado como 1.2 ou superior e o parâmetro **CHAR-VARYING** for configurado como COLLAPSE, <xsd:whiteSpace value="collapse"/> será gerado para sequências.

Tipo de Dados C e C++	simpleType de Esquema
char[z]	"type":"string" "maxLength": z

Tipo de Dados C e C++	simpleType de Esquema
char16_t[<i>n</i>]	<p>No nível de mapeamento 4.0 e superior:</p> <pre>"type": "string" "maxLength": <i>n</i></pre> <p>No tempo de execução, CICS preenche o campo de estrutura de dados do aplicativo com dados UTF-16.</p>
char[8] Suportado no nível de mapeamento 3.0 e superior quando DATETIME=PACKED15	<pre>"type": "string" "format": "date-time"</pre> <p>O formato do registro de data e hora é definido pelo RFC3339.</p>
char short int long long long	<pre>"type": "integer", "minimum": - (<i>n</i> + 1), "maximum": <i>n</i></pre> <p>em que <i>n</i> é o valor máximo que pode ser representado pelo primitivo.</p>
unsigned char unsigned short unsigned int unsigned long unsigned long long	<pre>"type": "integer", "minimum": 0, "maximum": <i>n</i></pre> <p>em que <i>n</i> é o valor máximo que pode ser representado pelo primitivo.</p>
bool (somente C++)	<pre>"type": "boolean"</pre>
float Suportado no nível de mapeamento 1.2 e superior.	<pre>"type": "number", "format": "float"</pre> <p>Nota: A representação de dados do IBM Hexadecimal Floating Point (HFP) não é exatamente igual à representação IEEE-754-1985 usada para JSON. Alguns valores podem não ser convertidos exatamente de uma representação para a outra. Alguns valores extremamente grandes ou pequenos podem não ser válidos para tipos de dados flutuantes. Alguns valores podem perder a precisão quando convertidos para ou a partir de uma representação HFP. Se conversões precisas forem importantes, considere substituir o uso de tipos de dados flutuantes por alternativas de precisão fixas.</p>
duplo Suportado no nível de mapeamento 1.2 e superior.	<pre>"type": "number", "format": "double"</pre> <p>Nota: A representação de dados do IBM Hexadecimal Floating Point (HFP) não é exatamente igual à representação IEEE-754-1985 usada para JSON. Alguns valores podem não ser convertidos exatamente de uma representação para a outra. Alguns valores extremamente grandes ou pequenos podem não ser válidos para tipos de dados duplo. Alguns valores podem perder a precisão quando convertidos para ou a partir de uma representação HFP. Se conversões precisas forem importantes, considere substituir o uso de tipos duplos de dados por alternativas de precisão fixas.</p>

Tipo de Dados C e C++	simpleType de Esquema
<i>type name [n]</i>	<p>Para primitivas:</p> <pre> "type": "array" "maxItems": n "minItems": n "items": { "tipo": "objeto", "properties": { name : { type JSON } } "required": [name] } </pre> <p>Para estruturas:</p> <pre> "type": "array" "maxItems": n "minItems": n "items": { type JSON } </pre> <p>Em que <i>type JSON</i> é a representação de esquema JSON do tipo C ou C++.</p>

Esquema JSON para mapeamento C e C++:

Os programas utilitários DFHJS2LS suportam mapeamentos entre os esquemas JSON e os tipos de dados C e C++.

Os assistentes do CICS geram nomes de campo válidos e exclusivos para variáveis C e C++ a partir dos nomes de elemento de esquema usando as regras a seguir:

1. Caracteres diferentes de A-Z, a-z, 0-9 ou _ são substituídos por ' X' inicial.
Por exemplo, monthly-total se torna monthlyXtotal.
2. Se o primeiro caractere não for um caractere alfabético, ele será substituído por um ' X' inicial.
Por exemplo, _monthlysummary se torna Xmonthlysummary.
3. Nomes duplicados no mesmo escopo se tornam exclusivos pela adição de um ou dois dígitos numéricos na segunda instância e nas instâncias subsequentes do nome.
Por exemplo, três instâncias de year se tornam year , year1 e year2.
4. Um esquema JSON especifica que uma variável possui variação de cardinalidade se possui um valor de "type" igual a "array" e as palavras-chave "minItems" e "maxItems" foram omitidas ou possuem valores diferentes. Se o esquema especifica que a variável possui variação de cardinalidade, os nomes de campo são criados com sufixos "_cont" e "_num".
Para obter mais informações, consulte “Matrizes variáveis de elementos no DFHJS2LS” na página 282.

5. Um esquema JSON especifica que uma variável é opcional se ela não aparece na matriz de palavra-chave "required" que está associada ao tipo de "object" de esquema JSON de inclusão. Para campos opcionais, um campo adicional é gerado com um sufixo _num incluído no nome de elemento. No tempo de execução, ele é zero para indicar que o valor estava ausente nos dados JSON e não zero se o valor estava presente nos dados JSON.
6. Os nomes de campo são limitados a 50 caracteres. Se um nome gerado, incluindo qualquer prefixo e sufixo, exceder esse comprimento, o nome de elemento será truncado.

O DFHJS2LS mapeia os valores de tipo de esquema JSON para tipos de dados C e C++ de acordo com a tabela a seguir. As regras a seguir também se aplicam:

- Se o parâmetro **MAPPING-LEVEL** for configurado como 1.2 ou superior e o parâmetro **CHAR-VARYING** for configurado como NULL, os dados de caractere de comprimento variável serão mapeados para sequências com terminação nula e um caractere extra será alocado para o terminador NULL.
- Se o parâmetro **MAPPING-LEVEL** for configurado como 1.2 ou superior e o parâmetro **CHAR-VARYING** for configurado como YES, os dados de caractere de comprimento variável serão mapeados para dois elementos relacionados: um campo de comprimento e um campo de dados.

Palavra-chave do Esquema JSON	Tipo de Dados C e C++
Todos de: "type": "null" "type": [] "enum": [] "allOf" "anyOf" "noneOf" "not"	Não-suportado
"\$schema": "http://json-schema.org/draft-04/schema#"	Esta palavra-chave é ignorada, mas assume-se que ela é compatível com a especificação de Esquema JSON 04 de rascunho.
"title": "same text" "description": "more text"	Essas palavras-chave são ignoradas.
"format": "<predefined values>"	A palavra-chave "format" é utilizada para modificar a estrutura gerada ou o valor de tempo de execução. Consulte as informações a seguir para o uso suportado de "format".

Palavra-chave do Esquema JSON	Tipo de Dados C e C++
<pre>"type": "array", "items": {<JSON Sub-schema>}, "additionalItems": false, "maxItems": m , "minItems": n</pre>	<p>A única forma de matriz JSON suportada atualmente é um número repetido dos mesmos valores de tipo. O <JSON Sub-schema> deve definir um "type" suportado , mas esse "type" não pode ser "array". Esta é uma restrição na estrutura de linguagem gerada.</p> <p>"additionalItems" é assumido como false e nenhum outro valor é suportado.</p> <p>Se "minItems" e "maxItems" estiverem presentes e forem iguais, a matriz será tratada como cardinalidade fixa, caso contrário, será tratada como cardinalidade variável. Consulte “Matrizes variáveis de elementos no DFHJS2LS” na página 282.</p>
<pre>"type": "array", "uniqueItems": true</pre>	<p>"uniqueItems" não é suportado com matrizes JSON. O <JSON Sub-schema> deve definir um "type" suportado , mas esse "type" não pode ser "array" . Esta é uma restrição na estrutura de linguagem gerada.</p>
<pre>"type": "object", "additionalProperties": false, "properties": { ["<element name>": {<JSON Sub-schema>} [,]]* } "required": [["<element name>" [,]]*]</pre>	<p>A única forma de objeto JSON atualmente suportada é um conjunto fixo de elementos nomeados.</p> <p>Isso gerará uma estrutura (ou subestrutura) usando os nomes de elementos.</p> <p>"additionalProperties" é assumido como false e nenhum outro valor é suportado.</p> <p>Qualquer elemento no objeto "properties" é considerado "optional" se ele não está na matriz "required" ou se nenhuma matriz "required" existe. Um elemento "optional" recebe uma cardinalidade variável de zero a X; em que X é 1 ou o número máximo de itens na matriz no qual esse item é definido como uma matriz. Consulte “Matrizes variáveis de elementos no DFHJS2LS” na página 282.</p>
<pre>"type": "object", "maxProperties": m, "minProperties": n, "patternProperties": {}, "dependencies":</pre>	<p>Nenhuma dessas palavras-chave é suportada com objetos JSON.</p>

Palavra-chave do Esquema JSON	Tipo de Dados C e C++
<pre>"type": "string" "maxLength": m "pattern": "regular expression>", "minLength": l</pre>	<p>char[z]</p> <p>em que o valor de z é baseado em m, porém é dependente das configurações do parâmetro CHAR-VARYING.</p> <p>m é baseado na palavra-chave "maxLength" e tratado como uma sequência de comprimento fixo.</p> <p>As restrições "pattern" e "minLength" são transmitidas para a estrutura de linguagem somente como um comentário.</p>
<pre>"type": "string" "maxLength": m</pre>	<p>Quando CCSID=1200 no nível de mapeamento 4.0 e superior:</p> <p>char16_t[z]</p> <p>em que o valor de z é baseado em m, porém é dependente das configurações do parâmetro CHAR-VARYING.</p> <p>m é baseado na palavra-chave "maxLength" e tratado como uma sequência de comprimento fixo.</p>
<pre>"*name*":{ "type": "string", "format": "date-time" }</pre>	<p>char[8]</p> <p>Todos suportados quando DATETIME=PACKED15</p>
<pre>"*name*":{ "type": "string", "format": "uri" }</pre>	<p>char[m]</p> <p>em que m é baseado na palavra-chave "maxLength" e tratado como uma sequência de comprimento fixo.</p> <p>Quando CCSID=1200 no nível de mapeamento 4.0 e superior:</p> <p>char16_t[m]</p> <p>em que m é baseado na palavra-chave "maxLength" e tratado como uma sequência de comprimento fixo.</p>
<pre>"*name*":{ "type": "string", "format": "base64Binary" }</pre>	<p>char[m]</p> <p>em que m é baseado na palavra-chave "maxLength".</p>
<pre>"*name*":{ "type": "string", "format": "hexBinary" }</pre>	<p>char[m]</p> <p>em que m é baseado na palavra-chave "maxLength".</p>

Palavra-chave do Esquema JSON	Tipo de Dados C e C++
<pre>"*name*":{ "type": "string", "format": "<predefined>" }</pre>	<p>char[<i>m</i>]</p> <p>em que <i>m</i> é baseado na palavra-chave "maxLength" e tratado como sequência de comprimento fixo e em que <predefined> é um de: <i>email</i> , <i>hostname</i> , <i>ipv4</i> ou <i>ipv6</i>. Uma "pattern" relevante é usado e transmitido ao comentário.</p> <p>Quando CCSID=1200 no nível de mapeamento 4.0 e superior:</p> <p>char16_t[<i>m</i>]</p> <p>em que <i>m</i> é baseado na palavra-chave "maxLength" e tratado como sequência de comprimento fixo e em que <predefined> é um de: <i>e-mail</i> , <i>hostname</i> , <i>ipv4</i> ou <i>ipv6</i> . Um "padrão" relevante é usado e transmitido ao comentário.</p>
"type": "boolean"	<p>bool (somente C++)</p> <p>short (somente C)</p>
<pre>"type": "integer", "maxExclusive": true, "minExclusive": true, "multipleOf": n</pre>	<p>As restrições "maxExclusive" e "minExclusive" são transmitidas à estrutura de linguagem somente como um comentário.</p> <p>"multipleOf" é ignorado.</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

Palavra-chave do Esquema JSON	Tipo de Dados C e C++
"type": "integer", minimum:0, maximum:18446744073709551615	unsigned long long
"type": "number", "maximum": m, "minimum": n, "maxExclusive": true, "minExclusive": true, "multipleOf": n	As restrições "maximum", "minimum", "maxExclusive" e "minExclusive" são transmitidas à estrutura de linguagem somente como um comentário. "multipleOf" é ignorado.
"type": "number" "format": "float"	Nível de mapeamento 1.1 e inferior: <ul style="list-style-type: none"> char[32] Nível de mapeamento 1.2 e posterior: <ul style="list-style-type: none"> float(*) <p>Nota: A representação de dados do IBM Hexadecimal Floating Point (HFP) não é exatamente igual à representação IEEE-754-1985 usada para JSON. Alguns valores podem não ser convertidos exatamente de uma representação para a outra. Alguns valores extremamente grandes ou pequenos podem não ser válidos para tipos de dados flutuantes. Alguns valores podem perder a precisão quando convertidos para ou a partir de uma representação HFP. Se conversões precisas forem importantes, considere substituir o uso de tipos de dados flutuantes por alternativas de precisão fixas.</p>
"type": "number" "format": "double"	Nível de mapeamento 1.0 e inferior: <ul style="list-style-type: none"> char[32] Nível de mapeamento 1.2 e posterior: <ul style="list-style-type: none"> double(*) <p>Nota: A representação de dados do IBM Hexadecimal Floating Point (HFP) não é exatamente igual à representação IEEE-754-1985 usada para JSON. Alguns valores podem não ser convertidos exatamente de uma representação para a outra. Alguns valores extremamente grandes ou pequenos podem não ser válidos para tipos de dados duplo. Alguns valores podem perder a precisão quando convertidos para ou a partir de uma representação HFP. Se conversões precisas forem importantes, considere substituir o uso de tipos duplos de dados por alternativas de precisão fixas.</p>

Nota: O CICS não pode transformar valores de número inteiro maiores que o valor máximo para um longo sinalizado ($2^{63} - 1$) a menos que eles sejam colocados entre aspas.

Nota: Os valores mínimo e máximo especificados no esquema para tipos numéricos são usados somente para mapear para um tipo de dados C ou C++. Os dados não são validados com relação a esses valores no tempo de execução.

Mapeamento de esquema PL/I para JSON:

O programa utilitário DFHLS2JS suporta mapeamentos entre as estruturas de dados PL/I e as definições de esquema JSON. Como o compilador Enterprise PL/I e os compiladores PL/I mais antigos diferem, duas opções de linguagem são suportadas: PLI-ENTERPRISE e PLI-OTHER.

Os nomes PL/I são convertidos em nomes JSON de acordo com as regras a seguir:

1. Os caracteres que não são válidos em nomes de propriedades JSON são substituídos por ' x '.
Por exemplo, `monthly$total` se torna `monthlyxtotal`.
2. Nomes duplicados se tornam exclusivos pela adição de um ou mais dígitos numéricos.
Por exemplo, duas instâncias de `year` se tornam `year` e `year1`.

O DFHLS2JS mapeia tipos de dados PL/I para elementos de esquema, de acordo com a tabela a seguir. Os tipos PL/I que não são mostrados na tabela não são suportados pelo DFHLS2JS. As restrições a seguir também se aplicam:

- Os itens de dados com o atributo COMPLEX não são suportados.
- Os itens de dados com o atributo FLOAT são suportados em um nível de mapeamento 1.2 ou superior. Enterprise PL/I FLOAT IEEE não é suportado.
- As sequências DBCS puras VARYING e VARYINGZ são suportadas em um nível de mapeamento 1.2 ou superior.
- Os itens de dados que são especificados como DECIMAL(*p* , *q*) são suportados somente quando $p \geq q$
- Os itens de dados que são especificados como BINARY(*p* , *q*) são suportados somente quando $q = 0$.
- Se o atributo PRECISION for especificado para um item de dados, ele será ignorado.
- As sequências PICTURE não são suportadas.
- Itens de dados ORDINAL são tratados como tipos de dados FIXED BINARY(7) .
- Se o parâmetro **MAPPING-LEVEL** estiver configurado como 1.2 ou superior e o parâmetro **CHAR-VARYING** estiver configurado como NULL, as matrizes de caracteres serão mapeadas para um string e serão processadas como sequências com terminação nula; este mapeamento não se aplica para Enterprise PL/I.
- Se o parâmetro **MAPPING-LEVEL** estiver configurado como 1.2 ou superior e o parâmetro **CHAR-VARYING** estiver configurado como BINARY, as matrizes de caracteres serão mapeadas para string e serão processadas como dados binários.
- Se o parâmetro **MAPPING-LEVEL** estiver configurado como 1.2 ou superior e o parâmetro **CHAR-VARYING** estiver configurado como COLLAPSE, os espaços em branco inicial e final serão removidos e vários espaços serão substituídos por um único espaço.

O DFHLS2JS não implementa completamente os algoritmos de preenchimento de PL/I; portanto, você deve declarar os bytes de preenchimento explicitamente em sua estrutura de dados. O DFHLS2JS emite uma mensagem se detecta que bytes de preenchimento estão ausentes. Cada estrutura de nível superior deve iniciar em um limite de palavra dupla e cada byte na estrutura deve ser mapeado para o limite correto. Considere este fragmento de código:

```
3 FIELD1 FIXED BINARY(7),
3 FIELD2 FIXED BINARY(31),
3 FIELD3 FIXED BINARY(63);
```

Neste exemplo:

- FIELD1 possui 1 byte de comprimento e pode ser alinhado em qualquer limite.
- FIELD2 tem 4 bytes de comprimento e deve ser alinhado em um limite de palavra inteira.
- FIELD3 tem 8 bytes de comprimento e deve ser alinhado em um limite de palavra dupla.

O compilador Enterprise PL/I alinha os campos na ordem a seguir:

1. FIELD3 é alinhado primeiro porque ele tem os requisitos de limite mais fortes.
2. FIELD2 é alinhado no limite de palavra inteira imediatamente antes de FIELD3.
3. FIELD1 é alinhado no limite de byte imediatamente antes de FIELD3.

Finalmente, para que a estrutura inteira esteja alinhada em um limite de palavra inteira, o compilador insere três bytes de preenchimento imediatamente antes de FIELD1.

Como DFHLS2JS não insere bytes de preenchimento equivalentes, você deve declará-los explicitamente antes da estrutura ser processada por DFHLS2JS. Por exemplo:

```
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);
```

Como alternativa, é possível mudar a estrutura para declarar todos os campos como desalinhados e recompilar o aplicativo que usa a estrutura. Para obter mais informações sobre requisitos de alinhamento de memória estrutural PL/I, consulte *Referência de Linguagem Enterprise PL/I*.

Descrição de Dados PL/I	Definição de esquema JSON
FIXED BINARY (<i>n</i>)	<pre>"type": "integer", "minimum":- (<i>n</i> + 1), "maximum": <i>n</i></pre> <p>em que <i>n</i> é o valor máximo que pode ser representado pelo primitivo.</p>
UNSIGNED FIXED BINARY(<i>n</i>) Restrição: Somente PL/I c Corporativo	<pre>"type": "integer", "minimum":0, "maximum": <i>n</i></pre> <p>em que <i>n</i> é o valor máximo que pode ser representado pelo primitivo.</p>

Descrição de Dados PL/I	Definição de esquema JSON
FIXED DECIMAL(<i>n</i> , <i>m</i>)	<pre>"type":"number", "format":"decimal", "multipleOf": x , "maximum": y , "mínimo":- z</pre> <p>em que:</p> <p><i>x</i> é a menor unidade disponível = $1/10^m$</p> <p><i>y</i> é o valor máximo que pode ser representado pela combinação de <i>n</i> e <i>m</i></p> <p><i>z</i> é o valor máximo que pode ser representado pela combinação de <i>n</i> e <i>m</i></p>
FIXED DECIMAL(15) Suportado no nível de mapeamento 3.0 e superior quando DATETIME=PACKED15	<pre>"type": "string", "format": "date-time"</pre> <p>O formato do registro de data e hora é definido pelo RFC3339.</p>
BIT(<i>n</i>) em que <i>n</i> é um múltiplo de 8. Outros valores não são suportados.	<pre>"type": "string" "maxLength": m</pre> <p>em que $m = n / 8$</p>
CHARACTER(<i>n</i>) VARYING e VARYINGZ também são suportados no nível de mapeamento 1.2 e superior. Restrição: VARYINGZ é suportado somente pelo PL/I Corporativo	<pre>"type": "string" "maxLength": n</pre>
GRAPHIC(<i>n</i>) VARYING e VARYINGZ também são suportados no nível de mapeamento 1.2 e superior. Restrição: VARYINGZ é suportado somente pelo Enterprise PL/I	<p>Em um nível de mapeamento 1.0 e 1.1, em que $m = 2 * n$:</p> <pre>"type": "string" "maxLength": m</pre> <p>Em um nível de mapeamento de 1.2 ou superior:</p> <pre>"type": "string" "maxLength": n</pre>

Descrição de Dados PL/I	Definição de esquema JSON
<p>WIDECHAR(<i>n</i>)</p> <p>Restrição: Somente PL/I c Corporativo</p>	<p>Em um nível de mapeamento 1.0 e 1.1, em que $m = 2 * n$:</p> <pre>"type": "string" "maxLength": m</pre> <p>Em um nível de mapeamento de 1.2 ou superior:</p> <pre>"type": "string" "maxLength": n</pre> <p>No nível de mapeamento 4.0 e superior, o CICS preenche o campo de estrutura de dados do aplicativo com dados 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>Restrição: Apenas PL/I corporativo</p>	<pre>"type": "integer", "minimum": 0, "maximum": 255</pre>
<p>BINARY FLOAT(<i>n</i>)</p> <p>em que $n \leq 21$</p> <p>Suportado no nível de mapeamento 1.2 e superior.</p> <p>Nota: A representação de dados do IBM Hexadecimal Floating Point (HFP) não é exatamente igual à representação IEEE-754-1985 usada para JSON. Alguns valores podem não ser convertidos exatamente de uma representação para a outra. Alguns valores extremamente grandes ou pequenos podem não ser válidos para tipos de dados flutuantes. Alguns valores podem perder a precisão quando convertidos para ou a partir de uma representação HFP. Se conversões precisas forem importantes, considere substituir o uso de tipos de dados BINARY FLOAT por alternativas de precisão fixas.</p>	<pre>"type": "number", "format": "float"</pre>

Descrição de Dados PL/I	Definição de esquema JSON
<p>BINARY FLOAT(<i>n</i>)</p> <p>em que $21 < n \leq 53$</p> <p>Valores maiores que 53 não são suportados.</p> <p>Suportado no nível de mapeamento 1.2 e superior.</p> <p>Nota: A representação de dados do IBM Hexadecimal Floating Point (HFP) não é exatamente igual à representação IEEE-754-1985 usada para JSON. Alguns valores podem não ser convertidos exatamente de uma representação para a outra. Alguns valores extremamente grandes ou pequenos podem não ser válidos para tipos de dados flutuantes. Alguns valores podem perder a precisão quando convertidos para ou a partir de uma representação HFP. Se conversões precisas forem importantes, considere substituir o uso de tipos de dados BINARY FLOAT por alternativas de precisão fixas.</p>	<pre>"type": "number", "format": "double"</pre>
<p>DECIMAL FLOAT(<i>n</i>)</p> <p>em que $n \leq 6$</p> <p>Suportado no nível de mapeamento 1.2 e superior.</p> <p>Nota: A representação de dados do IBM Hexadecimal Floating Point (HFP) não é exatamente igual à representação IEEE-754-1985 usada para JSON. Alguns valores podem não ser convertidos exatamente de uma representação para a outra. Alguns valores extremamente grandes ou pequenos podem não ser válidos para tipos de dados flutuantes. Alguns valores podem perder a precisão quando convertidos para ou a partir de uma representação HFP. Se conversões precisas forem importantes, considere substituir o uso de tipos de dados DECIMAL FLOAT por alternativas de precisão fixas.</p>	<pre>"type": "number", "format": "float"</pre>

Descrição de Dados PL/I	Definição de esquema JSON
<p>DECIMAL FLOAT(<i>n</i>)</p> <p>em que $6 < n \leq 16$</p> <p>Valores maiores que 16 não são suportados.</p> <p>Suportado no nível de mapeamento 1.2 e superior.</p> <p>Nota: A representação de dados do IBM Hexadecimal Floating Point (HFP) não é exatamente igual à representação IEEE-754-1985 usada para JSON. Alguns valores podem não ser convertidos exatamente de uma representação para a outra. Alguns valores extremamente grandes ou pequenos podem não ser válidos para tipos de dados flutuantes. Alguns valores podem perder a precisão quando convertidos para ou a partir de uma representação HFP. Se conversões precisas forem importantes, considere substituir o uso de tipos de dados DECIMAL FLOAT por alternativas de precisão fixas.</p>	<pre>"type": "number", "format": "double"</pre>
<p><i>name (n) data description</i></p>	<p>Para primitivas:</p> <pre>"type": "array" "maxItems": n "minItems": n "items": { "tipo": "objeto", "properties": { name : { data description JSON } } "required": [name] }</pre> <p>Para declarações de dados:</p> <pre>"type": "array" "maxItems": n "minItems": n "items": { data description JSON }</pre> <p>Em que <i>data description JSON</i> é a representação de esquema JSON da descrição de dados PL/I.</p>

Esquema JSON para mapeamento de PL/I:

O programa utilitário DFHJS2LS suporta mapeamentos entre esquemas JSON e estruturas de dados PL/I. Como o compilador PL/I Corporativo e compiladores PL/I mais antigos diferem, duas opções de linguagem são suportadas: PLI-ENTERPRISE e PLI-OTHER.

Regras para mapear nomes de elemento de esquema para PL/I

Os assistentes do CICS geram nomes exclusivos e válidos para variáveis PL/I a partir dos nomes de elemento de esquema usando as regras a seguir:

1. Caracteres diferentes de A-Z, a-z, 0-9, @, #, _ ou \$ são substituídos por ' X' inicial.

Por exemplo, monthly-total se torna monthlyXtotal.

É possível usar o parâmetro **MAPPING-OVERRIDES** para mudar o modo como outros caracteres são manipulados. Por exemplo, se você configurar o valor **HYPHENS-AS-UNDERSCORES**, qualquer hífen no esquema JSON será convertido em um sublinhado em vez de um X. Por exemplo, monthly-total se torna monthly_total.

2. Nomes duplicados no mesmo escopo se tornam exclusivos pela adição de um ou dois dígitos numéricos na segunda instância e nas instâncias subsequentes do nome.

Por exemplo, três instâncias de year se tornam year , year1 e year2.

3. Um esquema JSON especifica que uma variável possui variação de cardinalidade se possui um valor de "type" igual a "array" e as palavras-chave "minItems" e "maxItems" foram omitidas ou possuem valores diferentes. Se o esquema especifica que a variável possui variação de cardinalidade, os nomes de campo são criados com sufixos "_cont" e "_num".

Para obter mais informações, consulte “Matrizes variáveis de elementos no DFHJS2LS” na página 282.

4. Um esquema JSON especifica que uma variável é opcional se ela não aparece na matriz de palavra-chave "required" que está associada ao tipo de "object" de esquema JSON de inclusão. Para campos opcionais, um campo adicional é gerado com um sufixo _num incluído no nome de elemento. No tempo de execução, ele é zero para indicar que o valor estava ausente nos dados JSON e não zero se o valor estava presente nos dados JSON.
5. Os nomes de campo são limitados a 31 caracteres. Se um nome gerado, incluindo qualquer prefixo e sufixo, exceder esse comprimento, o nome de elemento será truncado.

O comprimento total do nome resultante é 31 caracteres ou menos.

Regras para mapear tipos de esquema para PL/I

DFHJS2LS mapeia valores de tipo de esquema para tipos de dados PL/I de acordo com a tabela a seguir. Além disso, observe os seguintes pontos:

- Se o parâmetro **MAPPING-LEVEL** for configurado como 1.2 ou superior e o parâmetro **CHAR-VARYING** for configurado como NULL, os dados de caractere de comprimento variável serão mapeados para sequências com terminação nula e um caractere extra será alocado para o terminador NULL.
- Se o parâmetro **MAPPING-LEVEL** for configurado como 1.2 ou superior e o parâmetro **CHAR-VARYING** não for especificado, por padrão, os dados de caractere de comprimento variável serão mapeados para um tipo de dados VARYINGZ para Enterprise PL/I e para o tipo de dados VARYING para Outro PL/I.

- Dados binários de comprimento variável são mapeados para um tipo de dados VARYING se são menores que 32.768 bytes e para um contêiner se são maiores que 32.768 bytes.

Palavra-chave do Esquema JSON	Descrição de Dados PL/I
<p>Todos de:</p> <pre>"type": "null" "type": [] "enum": [] "allOf" "anyOf" "noneOf" "not"</pre>	Não-suportado
<pre>"\$schema": "http://json-schema.org/draft-04/schema#"</pre>	Esta palavra-chave é ignorada, mas assume-se que ela é compatível com a especificação de Esquema JSON 04 de rascunho.
<pre>"title": "same text" "description": "more text"</pre>	Essas palavras-chave são ignoradas.
<pre>"format": "<predefined values>"</pre>	A palavra-chave "format" é usada para modificar a estrutura gerada ou o valor de tempo de execução. Consulte as informações a seguir para o uso suportado de "format".
<pre>"type": "array", "items": {<JSON Sub-schema>}, "additionalItems": false, "maxItems": m , "minItems": n</pre>	<p>A única forma de matriz JSON suportada atualmente é um número repetido dos mesmos valores de tipo. O <JSON Sub-schema> deve definir um "type" suportado, mas esse "type" não pode ser "array". Esta é uma restrição na estrutura de linguagem gerada.</p> <p>"additionalItems" é assumido como false e nenhum outro valor é suportado.</p> <p>Se "minItems" e "maxItems" estiverem presentes e forem iguais, a matriz será tratada como cardinalidade fixa, caso contrário, será tratada como cardinalidade variável. Consulte "Matrizes variáveis de elementos no DFHJS2LS" na página 282.</p>
<pre>"type": "array", "uniqueItems": true</pre>	"uniqueItems" não é suportado com matrizes JSON. O <JSON Sub-schema> deve definir um "type" suportado, mas esse "type" não pode ser "array". Esta é uma restrição na estrutura de linguagem gerada.

Palavra-chave do Esquema JSON	Descrição de Dados PL/I
<pre>"type": "object", "additionalProperties": false, "properties": { ["<element name>": {<JSON Sub-schema> } [,]]* } "required": [["<element name>" [,]]*]</pre>	<p>A única forma de objeto JSON atualmente suportada é um conjunto fixo de elementos nomeados.</p> <p>Isso gerará uma estrutura (ou subestrutura) usando os nomes de elementos.</p> <p>"additionalProperties" é assumido como false e nenhum outro valor é suportado.</p> <p>Qualquer elemento no objeto "properties" é considerado "optional" se ele não está na matriz "required" ou se nenhuma matriz "required" existe. Um elemento "optional" recebe uma cardinalidade variável de zero a X; em que X é 1 ou o número máximo de itens na matriz no qual esse item é definido como uma matriz. Consulte “Matrizes variáveis de elementos no DFHJS2LS” na página 282.</p>
<pre>"type": "object", "maxProperties": m, "minProperties": n, "patternProperties": {}, "dependencies":</pre>	<p>Nenhuma dessas palavras-chave é suportada com objetos JSON.</p>
<pre>"type": "string" "maxLength": m "pattern": "regular expression", "minLength": l</pre>	<p>char[z]</p> <p>em que o valor de z é baseado em m, porém é dependente das configurações do parâmetro CHAR-VARYING.</p> <p>m é baseado na palavra-chave "maxLength" e tratado como uma sequência de comprimento fixo.</p> <p>As restrições "pattern" e "minLength" são transmitidas para a estrutura de linguagem somente como um comentário.</p>
<pre>"type": "string" "maxLength": m</pre>	<p>Quando CCSID=1200 no nível de mapeamento 4.0 e superior:</p> <p>WIDECHAR(z)</p> <p>em que o valor de z é baseado em m, porém é dependente das configurações do parâmetro CHAR-VARYING.</p> <p>m é baseado na palavra-chave "maxLength" e tratado como uma sequência de comprimento fixo.</p>

Palavra-chave do Esquema JSON	Descrição de Dados PL/I
<pre>"*name*":{ "type": "string", "format":"date-time" }</pre>	<p>FIXED DECIMAL (15,0)</p> <p>Todos suportados quando DATETIME=PACKED15</p>
<pre>"*name*":{ "type": "string", "format":"uri" }</pre>	<p>CHAR(<i>m</i>)</p> <p>em que <i>m</i> é baseado na palavra-chave "maxLength" e tratado como sequência de comprimento fixo.</p> <p>Quando CCSID=1200 no nível de mapeamento 4.0 e superior:</p> <p>WIDECHAR(<i>m</i>)</p> <p>em que <i>m</i> é baseado na palavra-chave "maxLength" e tratado como uma sequência de comprimento fixo.</p>
<pre>"*name*":{ "type": "string", "format":"base64Binary" }</pre>	<p>CHAR(<i>m</i>)</p> <p>em que <i>m</i> é baseado na palavra-chave "maxLength".</p>
<pre>"*name*":{ "type": "string", "format":"hexBinary" }</pre>	<p>CHAR(<i>m</i>)</p> <p>em que <i>m</i> é baseado na palavra-chave "maxLength".</p>
<pre>"*name*":{ "type": "string", "format":"<predefined>" }</pre>	<p>CHAR(<i>m</i>) em que <i>m</i> é baseado na palavra-chave "maxLength" e tratado como uma sequência de comprimento fixo e <predefined> é um de: <i>email</i>, <i>hostname</i>, <i>ipv4</i> ou <i>ipv6</i>. Um "pattern" relevante é transmitido ao comentário.</p> <p>Quando CCSID=1200 no nível de mapeamento 4.0 e superior:</p> <p>WIDECHAR(<i>m</i>)</p> <p>em que <i>m</i> é baseado na palavra-chave "maxLength" e tratado como sequência de comprimento fixo e em que <predefined> é um de: <i>e-mail</i>, <i>hostname</i>, <i>ipv4</i> ou <i>ipv6</i>. Um "padrão" relevante é usado e transmitido ao comentário.</p>

Palavra-chave do Esquema JSON	Descrição de Dados PL/I
"type": "boolean"	<p>Nível de mapeamento 1.1 e inferior:</p> <p>Enterprise PL/I SIGNED FIXED BINARY (7)</p> <p>Outro PL/I FIXED BINARY (7)</p> <p>Nível de mapeamento 1.2 e posterior:</p> <p>Enterprise PL/I BIT(7) BIT(1)</p> <p>Outro PL/I BIT(7) BIT(1)</p> <p>em que BIT(7) é fornecido para alinhamento e BIT(1) Contém o valor mapeado Booleano.</p>
"type": "integer", "maxExclusive": true, "minExclusive": true, "multipleOf": n	<p>As restrições "maxExclusive" e "minExclusive" são transmitidas à estrutura de linguagem somente como um comentário.</p> <p>"multipleOf" é ignorado.</p>
"type": "integer", minimum:-128, maximum:127	<p>Enterprise PL/I SIGNED FIXED BINARY (7)</p> <p>Outro PL/I FIXED BINARY (7)</p>
"type": "integer", minimum:0, maximum:255	<p>Enterprise PL/I UNSIGNED FIXED BINARY (8)</p> <p>Outro PL/I FIXED BINARY (8)</p>
"type": "integer", minimum:-32768, maximum:32767	<p>Enterprise PL/I SIGNED FIXED BINARY (15)</p> <p>Outro PL/I FIXED BINARY (15)</p>
"type": "integer", minimum:0, maximum:65535	<p>Enterprise PL/I UNSIGNED FIXED BINARY (16)</p> <p>Outro PL/I FIXED BINARY (16)</p>
"type": "integer", minimum:-2147483648, maximum:2147483647	<p>Enterprise PL/I SIGNED FIXED BINARY (31)</p> <p>Outro PL/I FIXED BINARY (31)</p>

Palavra-chave do Esquema JSON	Descrição de Dados PL/I
"type": "integer", minimum:0, maximum:4294967295	Nível de mapeamento 1.1 e inferior: Enterprise PL/I UNSIGNED FIXED BINARY(32) Nível de mapeamento 1.2 e posterior: Enterprise PL/I CHAR(<i>y</i>) em que <i>y</i> é um comprimento fixo menor do que 16 MB. Todos os níveis de mapeamento: Outro PL/I BIT(64)
"type": "integer", minimum:-9223372036854775808, maximum:9223372036854775807	Nível de mapeamento 1.1 e inferior: Enterprise PL/I SIGNED FIXED BINARY(63) Nível de mapeamento 1.2 e posterior: Enterprise PL/I CHAR(<i>y</i>) em que <i>y</i> é um comprimento fixo menor do que 16 MB. Todos os níveis de mapeamento: Outro PL/I BIT(64)
"type": "integer", minimum:0, maximum:18446744073709551615	Nível de mapeamento 1.1 e inferior: Enterprise PL/I UNSIGNED FIXED BINARY(64) Nível de mapeamento 1.2 e posterior: Enterprise PL/I CHAR(<i>y</i>) em que <i>y</i> é um comprimento fixo menor do que 16 MB. Outro PL/I BIT(64)
"type": "number" "description": "decimal"	FIXED DECIMAL(<i>n</i> , <i>m</i>)

Palavra-chave do Esquema JSON	Descrição de Dados PL/I
<pre>"type": "number" "description": "float"</pre>	<p>Níveis de Mapeamento 1.0 e 1.1:</p> <ul style="list-style-type: none"> CHAR(32) <p>Nível de mapeamento 1.2 e posterior:</p> <p>Enterprise PL/I DECIMAL FLOAT(6) HEXADEC</p> <p>Outro PL/I DECIMAL FLOAT(6)</p> <p>Nota: A representação de dados do IBM Hexadecimal Floating Point (HFP) não é exatamente igual à representação IEEE-754-1985 usada para JSON. Alguns valores podem não ser convertidos exatamente de uma representação para a outra. Alguns valores extremamente grandes ou pequenos podem não ser válidos para tipos de dados flutuantes. Alguns valores podem perder a precisão quando convertidos para ou a partir de uma representação HFP. Se conversões precisas forem importantes, considere substituir o uso de tipos de dados DECIMAL FLOAT por alternativas de precisão fixas.</p>
<pre>"type": "number" "description": "double"</pre>	<p>Níveis de Mapeamento 1.0 e 1.1:</p> <ul style="list-style-type: none"> CHAR(32) <p>Nível de mapeamento 1.2 e posterior:</p> <p>Enterprise PL/I DECIMAL FLOAT(16) HEXADEC</p> <p>Outro PL/I DECIMAL FLOAT(16)</p> <p>Nota: A representação de dados do IBM Hexadecimal Floating Point (HFP) não é exatamente igual à representação IEEE-754-1985 usada para JSON. Alguns valores podem não ser convertidos exatamente de uma representação para a outra. Alguns valores extremamente grandes ou pequenos podem não ser válidos para tipos de dados flutuantes. Alguns valores podem perder a precisão quando convertidos para ou a partir de uma representação HFP. Se conversões precisas forem importantes, considere substituir o uso de tipos de dados DECIMAL FLOAT por alternativas de precisão fixas.</p>

Nota: O CICS não pode transformar valores de número inteiro maiores que o valor máximo para um longo sinalizado ($2^{63} - 1$) a menos que eles sejam colocados entre aspas.

Nota: Os valores mínimo e máximo especificados no esquema para tipos numéricos são usados somente para mapear para um tipo de dados PL/I. Os dados não são validados com relação a esses valores no tempo de execução.

Matrizes variáveis de elementos no DFHJS2LS

O JSON pode conter matrizes de números variados de elementos. Em geral, Esquemas JSON que contêm números variados de elementos não mapeiam eficientemente para uma única estrutura de dados de linguagem de alto nível. O CICS usa mapeamentos baseados em contêiner ou mapeamentos sequenciais para manipular números variados de elementos em dados JSON.

Uma matriz com um número variado de elementos é representada no esquema JSON usando as palavras-chave `minItems` e `maxItems` no esquema com valor `"type"` igual a `"array"` :

- A palavra-chave `minItems` especifica o número mínimo de vezes que o elemento pode ocorrer. Ela pode ter um valor 0 ou qualquer número inteiro positivo. Ela é padronizada com o valor 0.
- A palavra-chave `maxItems` especifica o número máximo de vezes que o elemento pode ocorrer. Ele pode ter um valor de qualquer número inteiro positivo maior ou igual ao valor da palavra-chave `minItems`.
- Se a palavra-chave `maxItems` está ausente, significa que a matriz é ilimitada.

Um campo opcional pode ser indicado por uma matriz variável de `"maxItems":1`. Por exemplo, uma sequência opcional 8 bytes chamada `"component"` :

```
"properties":{
  "component": {
    "type":"array",
    "maxItems":1,
    "items":{
      "type" : "string",
      "maxLength": 8
    }
  },
  "required": ["component"]
}
```

O mesmo efeito pode ser produzido não incluindo o nome do campo no valor da palavra-chave `"required"`:

```
"properties":{
  "component": {
    "type" : "string",
    "maxLength": 8
  }
}
```

Em geral, esquemas JSON que contêm números variados de elementos não mapeiam eficientemente para uma única estrutura de dados de linguagem de alto nível. Para lidar com esses casos, o CICS usa uma série de estruturas de dados conectadas que são transmitidas ao programa de aplicativo em uma série de contêineres. Essas estruturas são usadas como entrada e saída do aplicativo:

- Quando o CICS transforma dados JSON em dados do aplicativo, ele preenche estas estruturas com os dados do aplicativo e o aplicativo os lê.
- Quando o CICS transforma os dados do aplicativo em dados JSON, ele lê os dados do aplicativo nas estruturas que foram preenchidas pelo aplicativo.

Os exemplos a seguir ilustram o formato dessas estruturas de dados. Estes exemplos usam uma matriz de campos de 8 bytes simples. No entanto, o modelo suporta matrizes de tipos de dados complexos e matrizes de tipos de dados que contêm outras matrizes.

Exemplo 1. Número fixo de elementos

Este exemplo ilustra um elemento que ocorre exatamente três vezes:

```
"properties":{
  "component": {
    "type": "array",
    "maxItems": 3,
    "minItems": 3,
    "items":{
      "type" : "string",
      "maxLength": 8
    }
  },
  "required": ["component"]
```

Neste exemplo, o número de vezes que o elemento ocorre é conhecido antecipadamente, portanto, ele pode ser representado como uma matriz de comprimento fixo em uma declaração COBOL simples ou o equivalente em outras linguagens.

```
05 component PIC X(8) OCCURS 3 TIMES
```

Exemplo 2. Ativando o número de elementos no nível de mapeamento 2 e abaixo

Este exemplo ilustra um elemento obrigatório que pode ocorrer de uma a cinco vezes:

```
"properties":{
  "component": {
    "type": "array",
    "maxItems": 5,
    "minItems": 1,
    "items":{
      "type" : "string",
      "maxLength": 8
    }
  },
  "required": ["component"]
```

A estrutura de dados principal contém uma declaração de dois campos. Quando o CICS transforma os dados JSON em dados binários, o primeiro campo component-num contém o número de vezes que o elemento aparece nos dados JSON e o segundo campo, component-cont, contém o nome de um contêiner:

```
05 component-num PIC S9(9) COMP-5
05 component-cont PIC X(16)
```

Uma segunda estrutura de dados contém a declaração do elemento em si:

```
01 DFHJS-component
02 component PIC X(8)
```

Você deve examinar o valor de component-num, que conterá um valor no intervalo de 1 a 5, para descobrir quantas vezes o elemento ocorre. O conteúdo do elemento

está no contêiner nomeado em `component-cont`; o contêiner contém uma matriz de elementos, em que cada elemento é mapeado pela estrutura de dados `DFHJS-component`.

Se `minItems="0"`, ou está ausente, e `maxItems="1"`, o elemento é opcional. Para processar a estrutura de dados em seu programa de aplicativo, você deve examinar o valor de `component-num`:

- Se for zero, a mensagem não possui elemento do componente e o conteúdo de `component-cont` é indefinido.
- Se for um, o elemento do componente está no contêiner nomeado em `component-cont`.

O conteúdo do contêiner é mapeado pela estrutura de dados `DFHJS-component`.

Nota: Se os dados JSON consistirem em um único elemento recorrente, o `DFHJS2LS` gerará duas estruturas de linguagem. A estrutura de linguagem principal contém o número de elementos na matriz e o nome de um contêiner que contém a matriz de elementos. A segunda estrutura de linguagem mapeia uma única instância do elemento recorrente.

Exemplo 3. Número variado de elementos no nível de mapeamento 2.1 e superior

No nível de mapeamento 2.1 e acima, é possível usar o parâmetro **INLINE-MAXOCCURS-LIMIT** nos assistentes do CICS. O parâmetro **INLINE-MAXOCCURS-LIMIT** especifica a maneira como números variados de elementos são manipulados. As opções de mapeamento para números variados de elementos são o mapeamento baseado em contêiner, descrito em “Exemplo 2. Ativando o número de elementos no nível de mapeamento 2 e abaixo” na página 283 ou o mapeamento sequencial. O valor desse parâmetro pode ser um número inteiro positivo no intervalo de 0 a 32767:

- O valor padrão de **INLINE-MAXOCCURS-LIMIT** é 1, o qual assegura que elementos opcionais sejam mapeados sequencialmente.
- Um valor 0 para o parâmetro **INLINE-MAXOCCURS-LIMIT** evita o mapeamento sequencial.
- Se `maxItems` for menor ou igual ao valor de **INLINE-MAXOCCURS-LIMIT**, o mapeamento sequencial será usado.
- Se `maxItems` for maior do que o valor de **INLINE-MAXOCCURS-LIMIT**, o mapeamento baseado em contêiner será usado.

O mapeamento de números variados de elementos sequenciais resulta na geração de uma matriz, como acontece com o exemplo de ocorrência fixa acima, e de um contador. O campo `component-num` indica quantas instâncias do elemento estão presentes e elas são apontadas pela matriz. Para o exemplo mostrado em “Exemplo 2. Ativando o número de elementos no nível de mapeamento 2 e abaixo” na página 283, quando **INLINE-MAXOCCURS-LIMIT** é menor ou igual a 5, a estrutura de dados gerada conforme a seguir:

```
05 component-num PIC S9(9) COMP-5 SYNC.  
05 component OCCURS 5 PIC X(8).
```

O primeiro campo, `component-num`, é idêntico ao de saída para o mapeamento baseado em contêiner de exemplo na seção anterior. O segundo campo contém uma matriz de comprimento 5 que é grande o suficiente para conter o número máximo de elementos que podem ser gerados.

O mapeamento sequencial difere do mapeamento baseado em contêiner, que armazena o número de ocorrências do elemento e o nome do contêiner no qual os dados são colocados, porque ele armazena todos os dados no contêiner atual. O armazenamento dos dados no contêiner atual geralmente irá melhorar o desempenho e tornar o mapeamento sequencial preferível.

Exemplo 4. Matrizes variáveis aninhadas

Os esquemas JSON complexos podem conter elementos variáveis recorrentes, que que por sua vez contêm elementos variáveis recorrentes. Nesse caso, a estrutura descrita se estende além dos dois níveis descritos nos exemplos.

Este exemplo ilustra um elemento opcional chamado "component2" que está aninhado em um elemento obrigatório chamado "component1", onde o elemento obrigatório pode ocorrer de uma a cinco vezes:

```
"properties":{
  "component1": {
    "type": "array",
    "maxItems": 5,
    "minItems": 1,
    "items":{
      "type": "object",
      "properties":{
        "component2":{
          "type" : "string",
          "maxLength": 8
        }
      },
      "required": ["component2"]
    }
  },
  "required": ["component1"]
}
```

A estrutura de dados de nível superior é exatamente a mesma que nos exemplos anteriores:

```
05 component1-num PIC S9(9) COMP-5
05 component1-cont PIC X(16)
```

No entanto, a segunda estrutura de dados contém estes elementos:

```
01 DFHJS-component1
02 component2-num PIC S9(9) COMP-5
02 component2-cont PIC X(16)
```

Uma estrutura de terceiro nível contém estes elementos:

```
01 DFHJS-component2
02 component2 PIC X(8)
```

O número de ocorrências do elemento "component1" mais externo está em component1-num.

O contêiner nomeado em component1-cont contém uma matriz com esse número de instâncias da segunda estrutura de dados DFHJS-component1.

Cada instância de component2-cont nomeia um contêiner diferente, cada um dos quais contém a estrutura de dados mapeada pela estrutura de terceiro nível DFHJS-component2.

Para ilustrar essa estrutura, considere o fragmento de dados JSON que corresponde ao exemplo:

```

{"component1":
[
{
"component2": "string1"
},
{
"component2": "string2"
},
]
}

```

"component1" ocorre três vezes. Os dois primeiros contêm uma instância de "component2", a terceira instância não.

Na estrutura de dados de nível superior, component1-num contém um valor igual a 3. O contêiner nomeado em component1-cont possui três instâncias de DFHJS-component1 :

1. Na primeira, component2-num possui um valor 1 e o contêiner nomeado em component2-cont contém *string1*.
2. Na segunda, component2-num possui um valor 1 e o contêiner nomeado em component2-cont contém *string2*.
3. Na terceira, component2-num possui um valor 0 e o conteúdo de component2-cont é indefinido.

Neste exemplo, a estrutura de dados completa é representada por quatro contêineres:

- A estrutura de dados raiz no contêiner DFHJS-DATA.
- O contêiner nomeado em component1-cont.
- Dois contêineres nomeados nas duas primeiras instâncias de component2-cont.

Estruturas opcionais e a palavra-chave required

Estruturas de dados são definidas pelo Esquema JSON "type" de "object". Os esquemas relacionam nomes de campos a tipos individuais usando o objeto fornecido pela palavra-chave "properties". O requisito para esses campos fazerem parte dos dados JSON descritos pelo Esquema JSON é controlado pela matriz fornecida pela palavra-chave "required". Essa matriz lista todos os nomes de campos que devem estar presentes nos dados JSON. Os campos opcionais são, portanto, representados por sua ausência nessa matriz ou, como a matriz não tem permissão para estar vazia, a ausência da palavra-chave "required". Nesse caso, todos os campos são opcionais.

Os campos opcionais são tratados como uma matriz variável de 0 ou 1 elemento. Isso inclui um campo adicional com o sufixo "-num" anexado ao nome de elemento. Se o comprimento total é maior que 28 caracteres, o nome de elemento é truncado. No tempo de execução isso será diferente de zero para indicar que o valor estava presente nos dados JSON e zero se não estava.

Este exemplo mostra dois campos, um necessário chamado "required-structure" e o outro opcional chamado "optional-structure" :

```

{
"type": "object",
"properties": {
"required-structure": {
"type": "string",
"maxLength": 8
},

```

```

"optional-structure": {
  "type": "string",
  "maxLength": 8
},
"required": [
  "required-structure"
]
}

```

A estrutura COBOL gerada mostra ambos os campos, mas o segundo é precedido por "optional-structure-num" que é uma contagem de número inteiro dos elementos, com 0 representando nenhum e 1 indicando que ele está presente. O valor é configurado para indicar se o "optional-structure" contém dados válidos ou não.

```

03 OutputData.
06 required-structure PIC X(8).
06 optional-structure-num PIC S9(9) COMP-5 SYNC.
06 optional-structure PIC X(8).

```

Suporte para UTF-16 em dados do aplicativo

Os serviços da web CICS suportam a conversão de dados do aplicativo codificados em UTF-16 em XML ou JSON e também XML ou JSON em dados do aplicativo codificados em UTF-16. Use o UTF-16 quando você precisa armazenar e processar dados em diversos idiomas.

Os serviços da web CICS SOAP e JSON suportam a conversão de dados do aplicativo codificados em UTF-16 em XML ou JSON e também XML ou JSON em dados do aplicativo codificados em UTF-16. O Unicode é um esquema de codificação de largura variável que permite que os sistemas manipulem dados de forma eficiente.

UTF-16 é uma codificação de largura variável para Unicode, em que cada caractere é representado por 2 ou 4 bytes. Os serviços da web do CICS suportam CCSID 1200 para dados do aplicativo, que são UTF-16 BE (big endian) com a Área de Uso Privado da IBM. Esse comportamento é consistente com o suporte UTF-16 em todas as linguagens suportadas.

UTF-16 é suportado no nível de mapeamento 4.0 e superior. É possível customizar como os dados do aplicativo são convertidos usando configurações de mapeamento nos assistentes. Para obter mais informações sobre níveis de mapeamento XML, consulte Níveis de mapeamento para os assistentes CICS. Para obter mais informações sobre níveis de mapeamento JSON, consulte Níveis de mapeamento para os assistentes CICS JSON.

Nota: UTF-16 requer mais tempo de processamento e tem menos eficiência de armazenamento do que as codificações EBCDIC. Além disso, a combinação de tipos de codificação incorre em processamento de tempo de execução extra.

Mapeando UTF-16 do esquema XML ou JSON para estruturas de linguagem

O suporte para UTF-16 depende de como você cria o serviço da web. O mapeamento de esquema XML ou JSON para estruturas de linguagem, também conhecido como mapeamento de cima para baixo, tem as características a seguir. Se UTF-16 está ativado, todos os campos de texto são mapeados para campos UTF-16,

enquanto que tipos de dados de exibição numéricos em COBOL são mapeados como EBCDIC. Para usar UTF-16, configure o parâmetro CCSID de DFHJS2LS, DFHSC2LS ou DFHWS2LS para 1200.

Por exemplo, se o fragmento de esquema XML a seguir estiver presente no WSDL:

```
<xsd:element name="myString" nillable="false">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:maxLength value="20"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

O assistente DFHWS2LS gerará o campo a seguir em uma estrutura de linguagem COBOL:

```
myString PIC N(
20
) USAGE NATIONAL
```

O parâmetro CHAR-MULTIPLIER dos assistentes de serviços da web pode ser usado para especificar o comprimento de um campo que os assistentes geram.

CHAR-MULTIPLIER

Quando você usa UTF-16, os únicos valores válidos para o parâmetro **CHAR-MULTIPLIER** são 2 ou 4, em que 2 é o valor padrão.

CHAR-MULTIPLIER = 2, em que o esquema descreve uma sequência de $\text{maxlength} \times$, gera PIC N(x). Configurar **CHAR-MULTIPLIER** = 2 não impede o uso de pares substitutos em uma sequência UTF-16, mas afeta o número de caracteres que se ajustam no campo.

CHAR-MULTIPLIER = 4 gera PIC N($2x$). Se **CHAR-MULTIPLIER** = 4, o valor no tempo de execução é preenchido se a sequência inclui caracteres que podem ser expressos em uma única unidade de codificação.

Mapeando UTF-16 a partir de estruturas de linguagem para esquema XML ou JSON

O mapeamento de uma estrutura de linguagem para o esquema XML ou JSON, também conhecido como mapeamento bottom-up, é gerenciado de forma diferente do mapeamento de cima para baixo. Se uma sequência UTF-16 for declarada na estrutura de linguagem, os dados serão interpretados pelo CICS como codificados em UTF-16, caso contrário, assume-se que os dados estejam em uma codificação EBCDIC. O parâmetro CCSID para DFHLS2JS, DFHLS2SC ou DFHLS2WS indica a codificação de qualquer texto EBCDIC nos dados do aplicativo; ele não deve ser configurado para indicar UTF-16.

Os tipos de dados que são interpretadas como caracteres UTF-16 são os seguintes: PIC N(n) em COBOL, WIDECHAR(n) em PL/I e char16_t[n] em C e C++.

O parâmetro CHAR-USAGE dos assistentes de serviços da web pode ser usado para especificar os tipos de dados.

CHAR-USAGE

Em COBOL, o tipo de dado nacional, PIC N, pode ser usado para dados UTF-16 ou DBCS. Essa configuração é controlada pela opção do compilador NSYMBOL. Deve-se configurar o parâmetro **CHAR-USAGE** no

assistente com o mesmo valor que a opção do compilador NSYMBOL para assegurar que os dados sejam manipulados apropriadamente. Isso geralmente é configurado como CHAR-USAGE=NATIONAL quando você usa UTF-16.

Se você deseja combinar tipos de dados nacionais que contêm dados UTF-16 e DBCS no mesmo copybook, é possível usar os qualificadores USAGE NATIONAL ou USAGE DISPLAY-1 em campos individuais.

Nota: DFHLS2WS, DFHLS2SC e DFHLS2JS não suportam a cláusula COBOL GROUP USAGE NATIONAL.

Gerando mapeamentos a partir da estrutura de linguagem

Para criar JavaScript Object Notation (JSON) a partir de dados do aplicativo ou dados do aplicativo a partir de JSON, você cria os mapeamentos para descrever como o CICS deve transformar os dados e JSON no tempo de execução. É possível iniciar a partir de qualquer registro de dados do aplicativo; por exemplo, é possível iniciar com uma COMMAREA, arquivo VSAM, fila de armazenamento temporário ou um registro do IBM DB2.

Antes de Iniciar

Antes de criar os mapeamentos, você deve se certificar de que estas condições prévias foram atendidas:

- Deve-se ter uma estrutura de linguagem que descreva o registro de aplicativo em um conjunto de dados particionados. A estrutura de linguagem pode ser gravada em qualquer uma das linguagens de alto nível que são suportadas pelo assistente CICSJSON: COBOL, PL/I, C e C++.
- Deve-se configurar o ID do usuário sob o qual DFHLS2JS é executado para usar o z/OS UNIX.
- O ID do usuário deve ter permissão de leitura para acessar a estrutura de linguagem e permissão de gravação para colocar a saída nos diretórios apropriados no z/OS UNIX.
- Você deve alocar armazenamento suficiente para o ID do usuário para que o ID execute Java. É possível usar qualquer versão suportada de Java.

Sobre Esta Tarefa

Use o assistente JSON do CICS para criar os mapeamentos de dados para o registro de aplicativo. O assistente JSON do CICS cria pacote configurável do CICS e emite mensagens de erro sobre quaisquer itens não suportados que ele identifica em sua estrutura de linguagem. As informações de referência para o assistente CICS JSON listam as restrições que se aplicam a cada linguagem de alto nível.

Procedimento

Execute a tarefa em lote DFHLS2JS. DFHLS2JS possui parâmetros opcionais que você seleciona para atender aos seus requisitos, tal como selecionar uma página de códigos específica. Use os parâmetros a seguir como um mínimo:

- Especifique o idioma de alto nível de sua estrutura de linguagem no parâmetro **LANG**.
- Especifique o nome e o local de um recurso de pacote configurável no parâmetro **BUNDLE**.

- Especifique o nível de mapeamento no parâmetro **MAPPING-LEVEL**. Embora você possa usar qualquer nível de mapeamento, para obter as opções de mapeamento mais avançadas, use o nível de mapeamento mais recente.
- Especifique o local e a página de códigos das estruturas de linguagem que descrevem o registro de aplicativo nos parâmetros **PDSMEM** e **PDSCP**.
- Especifique o nome e o local do arquivo de esquema JSON .json no parâmetro **JSON-SCHEMA**. Se o arquivo não existir, o DFHLS2JS criará o esquema JSON, mas não a estrutura de diretório.
- Especifique o nome que é usado para o recurso de pacote configurável JSONTRANSFRM. Esse nome é usado por aplicativos para identificar os mapeamentos JSON.

A tarefa em lote cria uma estrutura de diretório de pacote configurável no z/OS UNIX. O diretório do pacote configurável possui um subdiretório META-INF que contém o manifest do pacote configurável. A tarefa em lote também cria um esquema JSON e uma ligação JSON no pacote configurável, usando os nomes de arquivo que você especificou para os parâmetros **JSONTRANSFRM** e **JSON-SCHEMA**.

Instale o recurso BUNDLE que especifica essa ligação JSON. O recurso de pacote configurável JSONTRANSFRM cria dinamicamente um recurso JSONTRANSFRM, que define o local do esquema JSON e do arquivo de ligação. É possível usar as visualizações operacionais do Pacote Configurável do CICS Explorer e das Partes do Pacote Configurável para verificar o status de recursos BUNDLE instalados.

Resultados

Um pacote configurável do CICS que contém uma transformação JSON é gerado.

O exemplo a seguir mostra DFHLS2JS com o conjunto mínimo de parâmetros 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
/*
```

O que Fazer Depois

Grave um programa de aplicativo para transformar os dados do aplicativo em JSON e vice-versa. É possível usar os mesmos mapeamentos para ambas as transformações.

Gerando mapeamentos a partir de um esquema JSON

Para criar dados do aplicativo a partir do JavaScript Object Notation (JSON) ou JSON a partir de dados do aplicativo, crie os mapeamentos para descrever como o CICS deve transformar os dados e o JSON no tempo de execução. É possível iniciar a partir de um esquema JSON. Após a estrutura de linguagem e o

mapeamento serem gerados, você pode desenvolver o aplicativo CICS usando a estrutura de linguagem e transformar JSON em dados do aplicativo e vice-versa.

Antes de Iniciar

Antes de criar os mapeamentos, você deve se certificar de que estas condições prévias foram atendidas:

- Deve-se ter o esquema JSON que descreve um registro JSON.
- Deve-se configurar o ID do usuário sob o qual o DFHJS2LS é executado para usar o z/OS UNIX.
- O ID do usuário deve ter permissão de leitura para acessar o esquema JSON e permissão de gravação para colocar a saída nos diretórios apropriados no z/OS UNIX.
- Você deve alocar armazenamento suficiente para o ID do usuário para que o ID execute Java. É possível usar qualquer versão suportada de Java.

Sobre Esta Tarefa

Use o assistente JSON do CICS para criar os mapeamentos de dados para o registro de aplicativo. O assistente CICS JSON cria um pacote configurável do CICS e emite mensagens de erro sobre quaisquer itens não suportados que identifica em sua estrutura de linguagem. As informações de referência para o assistente CICS JSON listam as restrições que se aplicam a cada linguagem de alto nível. Para obter mais informações, consulte “DFHJS2LS: Esquema JSON para conversão de linguagem de alto nível para interface vinculável” na página 418.

Procedimento

Execute a tarefa em lote DFHJS2LS. DFHJS2LS possui parâmetros opcionais que você seleciona para atender seus requisitos, tal como selecionar uma página de códigos específica. Use os parâmetros a seguir como um mínimo:

- Especifique o nome e o local de um recurso de pacote configurável no parâmetro **BUNDLE**.
- Especifique o nome e o local do arquivo de esquema JSON no parâmetro **JSON-SCHEMA**.
- Especifique o nível de mapeamento no parâmetro **MAPPING-LEVEL**. Embora você possa usar qualquer nível de mapeamento, para obter as opções de mapeamento mais avançadas, use o nível de mapeamento mais recente.
- Especifique a linguagem de alto nível para a estrutura de linguagem gerada no parâmetro **LANG**.
- Especifique o local e a página de códigos das estruturas de linguagem que descrevem o registro de aplicativo nos parâmetros **PDSMEM** e **PDSCP**. O DFHJS2LS cria a estrutura de linguagem, mas não a estrutura de diretório.
- Especifique o nome que é usado para o recurso do pacote configurável JSONTRANSFRM no CICS. Esse nome é usado por aplicativos para identificar os mapeamentos JSON.

A tarefa em lote cria uma estrutura de diretório de pacote configurável no z/OS UNIX. O diretório do pacote configurável possui um subdiretório META-INF que contém o manifest do pacote configurável. A tarefa em lote também cria uma ligação JSON e copia o esquema JSON no diretório do pacote configurável, usando os nomes de arquivo que você especificou para os parâmetros **JSONTRANSFRM** e **JSON-SCHEMA**. A tarefa em lote também cria a estrutura de linguagem no local especificado nos parâmetros **PDSMEM** e **PDSLIB**.

Instale o recurso BUNDLE que especifica essa ligação JSON. O recurso de pacote configurável JSONTRANSFRM cria dinamicamente um recurso JSONTRANSFRM, que define o local do esquema JSON e do arquivo de ligação. Esse recurso de pacote configurável fica visível no CICS quando você visualiza o conteúdo do pacote configurável instalado usando o CICS Explorer. Ele não é um recurso normal do CICS e não fica visível ao usar CEMT ou a WUI CPSM.

Resultados

Um pacote configurável do CICS que contém uma transformação JSON é gerado. Uma estrutura de linguagem é gerada.

O exemplo a seguir mostra DFHJS2LS com o conjunto mínimo de parâmetros 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
/*
```

O que Fazer Depois

Grave um programa de aplicativo para transformar os dados do aplicativo em JSON e vice-versa. É possível usar os mesmos mapeamentos para ambas as transformações.

Transformando dados do aplicativo em JSON vinculando-se ao DFHJSON

A interface vinculável do transformador JSON DFHJSON é um programa fornecido pelo CICS que pode ser chamado para executar a transformação entre dados do aplicativo e JSON. Seu programa de aplicativo pode transformar dados do aplicativo em JSON vinculando-se ao DFHJSON.

Antes de Iniciar

Deve-se ter um recurso de pacote configurável JSONTRANSFRM ativado que defina a ligação JSON e o esquema JSON. Se você pretende executar transformações usando Java, deve-se ter um servidor JVM Axis2 já em execução.

Sobre Esta Tarefa

Crie ou atualize um programa de aplicativo para vincular-se ao programa DFHJSON fornecido pelo CICS para fazer a transformação.

Procedimento

1. O programa de aplicativo deve criar um canal, por exemplo *MyChannelName* e colocar os contêineres a seguir no canal.
 - DFHJSON-DATA

- DFHJSON-TRANSFRM
- DFHJSON-JVMSERVER (opcional)

Para obter mais informações sobre estes contêineres, consulte Contêineres de interface vinculáveis do transformador JSON.

2. Use o comando **EXEC CICS LINK PROGRAM** para transformar os dados em JSON:
EXEC CICS LINK PROGRAM('DFHJSON') CHANNEL('MyChannelName')
3. Obtenha o contêiner DFHJSON-ERROR e verifique se algum erro ocorreu durante a transformação.
4. Obtenha o contêiner DFHJSON-JJSON e faça uso do JSON em seu aplicativo.
5. Instale o aplicativo.

Resultados

Quando o aplicativo executa o comando **LINK**, o CICS verifica o recurso do pacote configurável JSONTRANSFRM para localizar os mapeamentos na ligação JSON e transforma os dados binários do aplicativo em JSON usando os contêineres no canal. O JSON é colocado no contêiner DFHJSON-JJSON no retorno. O JSON está em conformidade com o esquema JSON que está definido no recurso do pacote configurável JSONTRANSFRM.

O que Fazer Depois

Também é possível usar os mesmos mapeamentos para transformar o JSON em dados do aplicativo. Para obter mais informações, consulte “Transformando JSON em dados do aplicativo vinculando-se ao DFHJSON” na página 483.

Transformando dados do aplicativo em JSON usando o comando de API TRANSFORM DATATOJSON

É possível usar o comando de API **TRANSFORM DATATOJSON** em seu aplicativo para transformar dados do aplicativo em JSON.

Antes de Iniciar

Deve-se ter um recurso JSONTRANSFRM ativado que defina a ligação JSON e o esquema JSON. Se você pretende executar transformações usando Java, deve-se ter um servidor JVM Axis2 já em execução.

Sobre Esta Tarefa

O aplicativo deve usar uma interface baseada em canal.

Procedimento

1. Crie um canal e coloque no canal um contêiner de entrada que contenha os dados do aplicativo a serem convertidos.

Nota: Este canal também terá um contêiner de saída que contém a saída JSON quando o comando **TRANSFORM DATATOJSON** for concluído. Não crie o contêiner de saída antes de emitir **TRANSFORM DATATOJSON** porque o contêiner é criado e preenchido como parte do próprio comando.

2. Use o comando **TRANSFORM DATATOJSON** para transformar os dados em JSON. Por exemplo:

```
EXEC CICS TRANSFORM DATATOJSON CHANNEL(
  ChannelName
) INCONTAINER(
  InpContainerName
) OUTCONTAINER(
  OutContainerName
) TRANSFORMER(
  BundleName
)
```

Resultados

Quando o aplicativo executa o comando **TRANSFORM DATATOJSON**, o CICS verifica o recurso do pacote configurável JSONTRANSFRM para localizar os mapeamentos na ligação JSON e transforma os dados binários do aplicativo em JSON usando os contêineres no canal. No retorno, o JSON é colocado no contêiner que é especificado na opção **OUTCONTAINER** do comando **TRANSFORM DATATOJSON**. Se a opção for omitida, DFHJSON-JSON será usado por padrão. O JSON está em conformidade com o esquema JSON que é definido no recurso do pacote configurável JSONTRANSFRM.

Transformando JSON em dados do aplicativo vinculando-se ao DFHJSON

A interface vinculável do transformador JSON DFHJSON é um programa fornecido pelo CICS que pode ser chamado para executar a transformação entre dados do aplicativo e JSON. Seu programa de aplicativo pode transformar o JSON em dados do aplicativo vinculando-se ao DFHJSON.

Antes de Iniciar

Deve-se ter um recurso JSONTRANSFRM ativado que defina a ligação JSON e o esquema JSON. Se você pretende executar transformações usando Java, deve-se ter um servidor JVM Axis2 já em execução.

Sobre Esta Tarefa

Crie ou atualize um programa de aplicativo para vincular-se ao programa DFHJSON fornecido pelo CICS para fazer a transformação.

Procedimento

1. Crie um canal, por exemplo *MyChannelName*, e coloque os contêineres a seguir no canal.
 - DFHJSON-JSON
 - DFHJSON-TRANSFRM
 - DFHJSON-JVMSERVER

Para obter mais informações sobre estes contêineres, consulte Contêineres de interface vinculáveis do transformador JSON.

2. Use o comando **EXEC CICS LINK PROGRAM** Comando de API para transformar os dados em JSON:

```
EXEC CICS LINK PROGRAM('DFHJSON') CHANNEL('MyChannelName')
```

3. Instale o programa de aplicativo.

Resultados

Quando o aplicativo executa o comando **LINK PROGRAM**, o CICS verifica o recurso JSONTRANSFRM para localizar os mapeamentos na ligação JSON e transforma o JSON nos dados do aplicativo usando os contêineres no canal. Os dados do aplicativo são colocados no contêiner de bits DFHJSON-DATA no retorno.

O que Fazer Depois

Também é possível usar os mesmos mapeamentos para transformar dados do aplicativo em JSON. Para obter mais informações, consulte “Transformando dados do aplicativo em JSON vinculando-se ao DFHJSON” na página 481.

Transformando JSON em dados do aplicativo usando o comando de API TRANSFORM JSONTODATA

É possível usar o comando de API **TRANSFORM JSONTODATA** em seu aplicativo para transformar o JSON em dados do aplicativo.

Antes de Iniciar

Deve-se ter um recurso JSONTRANSFRM ativado que defina a ligação JSON e o esquema JSON. Se você pretende executar transformações usando Java, deve-se ter um servidor JVM Axis2 já em execução.

Sobre Esta Tarefa

O aplicativo deve usar uma interface baseada em canal.

Procedimento

1. Crie um canal e coloque no canal um contêiner de entrada que contém o JSON a ser convertido.

Nota: Este canal também terá um contêiner de saída que contém os dados convertidos quando o comando **TRANSFORM JSONTODATA** é concluído. Não crie o contêiner de saída antes de emitir **TRANSFORM JSONTODATA** porque o contêiner é criado e preenchido como parte do próprio comando.

2. Use o comando **TRANSFORM JSONTODATA** para transformar o JSON em dados do aplicativo. Por exemplo:

```
EXEC CICS TRANSFORM JSONTODATA CHANNEL(  
  ChannelName  
) INCONTAINER(  
  InpContainerName  
) OUTCONTAINER(  
  OutContainerName  
) TRANSFORMER(  
  BundleName  
)
```

Resultados

Quando o aplicativo executa o comando **TRANSFORM JSONTODATA**, o CICS verifica o recurso de pacote configurável JSONTRANSFRM para localizar os mapeamentos na ligação JSON e transforma o JSON nos dados binários do aplicativo usando os contêineres no canal. No retorno, os dados convertidos são colocados no contêiner que é especificado na opção **OUTCONTAINER** do comando **TRANSFORM JSONTODATA**. Se a opção for omitida, DFHJSON-DATA será usado por padrão.

Criando um aplicativo cliente de serviço da web JSON

É possível gravar um programa de aplicativo para chamar um serviço da web RESTful usando a interface vinculável para transformar JSON e, em seguida, usar os comandos WEB para enviá-lo para o provedor de serviços remotos.

Antes de Iniciar

Deve-se estar familiarizado com o uso da interface vinculável para transformar o JSON, conforme descrito em Transformando dados do aplicativo e JSON usando a interface vinculável e com as APIs EXEC CICS WEB, conforme descrito em .

Sobre Esta Tarefa

Como parte de um aplicativo do CICS, você pode desejar chamar um serviço da web RESTful hospedado em outro sistema. Para fazer isso, você deve primeiro descrever os dados a serem trocados com o serviço remoto. É possível, então, gravar um programa de aplicativo que usa a API EXEC CICS WEB para se comunicar com o serviço remoto com o protocolo HTTP para enviar dados da solicitação para o serviço e receber dados de resposta. É possível usar a interface vinculável para transformar seus dados do aplicativo para JSON para usar como parte da solicitação e transformar a resposta JSON em dados do aplicativo. Alguns serviços podem não suportar uma carga útil para a solicitação e a resposta.

Procedimento

1. Defina a interface para o serviço remoto.
 - a. Se o serviço remoto existir, verifique se um esquema JSON que descreve as cargas úteis de solicitação e resposta está disponível. Se não, você deverá criar um. Em seguida, use o assistente JSON para gerar um mapeamento para uma estrutura de linguagem. Para obter mais informações, consulte “Gerando mapeamentos a partir de um esquema JSON” na página 479.
 - b. Se o serviço remoto ainda não existir e você desejar basear sua interface na estrutura de dados do aplicativo, use o assistente JSON para gerar um esquema JSON. Em seguida, transmita o esquema JSON para o desenvolvedor de aplicativos de serviço remoto. Para obter mais informações, consulte “Gerando mapeamentos a partir da estrutura de linguagem” na página 478.
2. Defina um recurso BUNDLE para o pacote configurável que é gerado pelo assistente JSON e instale o pacote configurável no CICS.
3. Defina um recurso URIMAP para o terminal em serviço remoto e instale-o. Para obter mais informações, consulte Recursos URIMAP.
4. Crie ou atualize um programa de aplicativo para chamar o serviço remoto, conforme a seguir:
 - a. Se o serviço remoto requerer uma carga útil JSON para a solicitação (por exemplo, quando o método HTTP é POST ou PUT), use a interface vinculável para transformar seus dados do aplicativo em JSON. Para obter mais informações, consulte “Transformando dados do aplicativo em JSON vinculando-se ao DFHJSON” na página 481.
 - b. Abra uma conexão com o servidor no qual o serviço remoto está hospedado, usando o comando **EXEC CICS WEB OPEN**. Para obter mais informações, consulte WEB OPEN.
 - c. Dependendo dos requisitos do serviço, você pode desejar codificar o comando **EXEC CICS WEB WRITE HTTPHEADER** para especificar o cabeçalho

Content-Type application/JSON para indicar que o JSON está sendo fornecido. Para obter mais informações, consulte WEB WRITE HTTPHEADER.

- d. Codifique um comando **EXEC CICS WEB CONVERSE** para enviar a solicitação para o serviço remoto e receber a resposta. Especifique a sequência de consultas ou o corpo da solicitação (a partir do contêiner DFHJSON-JSON), se necessário. Se você espera uma resposta do serviço, especifique o contêiner DFHJSON-JSON para receber o JSON de resposta. Para obter mais informações, consulte WEB CONVERSE.
- e. Se você não espera fazer solicitações adicionais, codifique um comando **EXEC CICS WEB CLOSE** para fechar a conexão. Para obter mais informações, consulte WEB CLOSE.
- f. Verifique o código de resposta HTTP que é retornado pelo comando **EXEC CICS WEB CONVERSE** e execute a ação apropriada se um erro ocorreu. Por exemplo, tente a solicitação novamente ou retorne um erro para o usuário.
- g. Se um corpo de resposta era esperado a partir do serviço remoto, use a interface vinculável para transformar o JSON em dados do aplicativo. Para obter mais informações, consulte “Transformando dados do aplicativo em JSON vinculando-se ao DFHJSON” na página 481.

Resultados

Você criou um aplicativo que pode chamar um serviço da web RESTful com uma carga útil JSON.

Criando um serviço da web SOAP

É possível expor aplicativos do CICS existentes como serviços da web SOAP e criar novos aplicativos do CICS para agir como provedores ou solicitantes de serviço da web SOAP.

Antes de Iniciar

Antes de começar a criar um serviço da web SOAP, execute estas tarefas:

1. Configure seu sistema CICS para suportar serviços da web; consulte “Configurando seu Sistema CICS para Serviços da Web” na página 39.
2. Crie a infraestrutura necessária para suportar a implementação de seus serviços da web; consulte “Criando a Infraestrutura de Serviços da Web” na página 52.
3. Decida se você deseja usar o assistente de serviços da web; consulte “Planejando para usar serviços da web SOAP” na página 26.

Sobre Esta Tarefa

O assistente de serviços da web do CICS é um utilitário fornecido que ajuda a criar os artefatos necessários para um novo provedor de serviço da web SOAP ou um aplicativo solicitante de serviço ou para ativar um aplicativo existente como um provedor de serviço da web.

O assistente de serviços da web do CICS pode criar um documento WSDL a partir de uma estrutura de linguagem simples ou uma estrutura de linguagem a partir de um documento WSDL existente; ele suporta COBOL, C/C++ e PL/I. Ele também gera informações que são usadas para ativar a conversão de tempo de execução

automática das mensagens SOAP para contêineres e COMMAREAs e vice-versa. Essas informações são usadas pelo suporte de serviços da web do CICS durante o processamento de pipeline.

Crie seu serviço da web, conforme descrito no procedimento a seguir e valide que ele funciona corretamente:

Procedimento

1. Crie um serviço da web SOAP de uma das quatro maneiras:
 - Use o assistente de serviços da web para criar a descrição de serviços da web ou as estruturas de linguagem e implementá-las no CICS. Use o comando **PIPELINE SCAN** para criar automaticamente os recursos necessários do CICS.
 - Use o IBM Developer for z Systems ou a API Java para criar a descrição de serviços da web ou as estruturas de linguagem e implementá-las no CICS. Use o comando **PIPELINE SCAN** para criar automaticamente os recursos do CICS.
 - Crie ou mude um programa de aplicativo para manipular o XML nas mensagens de entrada e de saída, incluindo a conversão de dados, e preencha os contêineres corretos no pipeline. Deve-se criar os recursos necessários do CICS manualmente.
 - Implemente um aplicativo Axis2 como um serviço da web.
2. Inicie o serviço da web para testar se ele funciona conforme desejado. Se você estiver usando o assistente de serviços da web para implementar seu serviço da Web, poderá usar o comando **SET WEBSERVICE** para ativar a validação. Essa validação verifica se os dados foram convertidos corretamente.

O que Fazer Depois

Essas etapas são explicadas em mais detalhes nos tópicos a seguir.

O assistente de serviços da web do CICS

O assistente de serviços da web do CICS é um conjunto de utilitários em lote que podem ajudá-lo a transformar aplicativos CICS existentes em serviços da web e a ativar os aplicativos CICS para usar os serviços da web fornecidos por provedores externos. O assistente suporta a implementação rápida de aplicativos CICS para uso em provedores de serviços e solicitantes de serviços, com o mínimo de esforço de programação.

Quando você usa o assistente de serviços da web para CICS, não é necessário gravar seu próprio código para analisar mensagens de entrada e para construir mensagens de saída; o CICS mapeia dados entre o corpo de uma mensagem SOAP e a estrutura de dados do programa de aplicativo.

O assistente pode criar um documento WSDL a partir de uma estrutura de linguagem simples ou de uma estrutura de linguagem a partir de um documento WSDL existente e suporta COBOL, C/C++ e PL/I. Ele também gera informações usadas para ativar a conversão automática de tempo de execução das mensagens SOAP para contêineres e COMMAREAs e vice-versa.

O assistente de serviços da web do CICS consiste em dois programas utilitários:

DFHLS2WS

Gera um arquivo de ligação de serviço da web a partir de uma estrutura de linguagem. Este utilitário também gera uma descrição de serviços da web.

DFHWS2LS

Gera um arquivo de ligação de serviço da web a partir de uma descrição de serviços da web. Este utilitário também gera uma estrutura de linguagem que pode ser usada em seus programas de aplicativo.

Os procedimentos JCL para executar ambos os programas estão na biblioteca *hlq*.XDFHINST .

Para obter mais informações sobre os programas utilitários e dados de mapeamentos do assistente de serviços da web, consulte os tópicos a seguir.

DFHLS2WS: Linguagem de Alto Nível para Conversão WSDL

O procedimento DFHLS2WS gera uma descrição de serviços da web e um arquivo de ligação de serviço da web a partir de uma estrutura de dados de linguagem de alto nível. Você pode utilizar DFHLS2WS quando expuser um programa aplicativo do CICS como um provedor de serviços.

As instruções de controle de tarefa para DFHLS2WS, seus parâmetros simbólicos, seus parâmetros de entrada, suas descrições e uma tarefa de exemplo o ajudam a usar este procedimento.

Instruções de Controle da Tarefa para DFHLS2WS

JOB Inicia a tarefa.

EXEC Especifica o nome do procedimento (DFHLS2WS).

INPUT.SYSUT1 DD

Especifica a entrada. Os parâmetros de entrada geralmente são especificados no fluxo de entrada. No entanto, eles podem ser definidos em um conjunto de dados ou em um membro de um conjunto de dados particionados.

Parâmetros simbólicos

Os parâmetros simbólicos a seguir são definidos no DFHLS2WS:

JAVADIR = *path*

Especifica o nome do diretório Java que é usado pelo DFHLS2WS. O valor desse parâmetro é anexado a */usr/lpp/* para produzir um nome de caminho completo de */usr/lpp/ path* .

Normalmente, você não especifica este parâmetro. O valor padrão é o valor que foi fornecido à tarefa de instalação do CICS (DFHISTAR) no parâmetro **JAVADIR**.

PATHPREF = *prefix*

Especifica um prefixo opcional que estende o caminho do diretório z/OS UNIX usado em outros parâmetros. O padrão é a cadeia vazia.

Normalmente, você não especifica este parâmetro. O valor padrão é o valor que foi fornecido para a tarefa de instalação do CICS (DFHISTAR) no parâmetro **PATHPREF**.

SERVICE = *value*

Use este parâmetro somente quando orientado a fazer isso pelo Suporte IBM.

TMPDIR = *tmpdir*

Especifica o local de um diretório no z/OS UNIX que o DFHLS2WS usa como uma área de trabalho provisória. O ID do usuário sob o qual a tarefa é executada deve ter permissão de leitura e gravação para esse diretório.

O valor padrão é /tmp.

TMPPFILE = *tmpprefix*

Especifica um prefixo que o DFHLS2WS usa para construir os nomes dos arquivos da área de trabalho provisória.

O valor padrão é LS2WS.

USSDIR = *path*

Especifica o nome do diretório do CICS TS no sistema de arquivos de serviços do sistema UNIX. O valor desse parâmetro é anexado a /usr/lpp/cicsts/ para produzir um nome de caminho completo de /usr/lpp/cicsts/*path*.

Normalmente, você não especifica este parâmetro. O valor padrão é o valor que foi fornecido à tarefa de instalação do CICS (DFHISTAR) no parâmetro **USSDIR**.

O Espaço de Trabalho Temporário

O DFHLS2WS cria os três arquivos temporários a seguir no tempo de execução:

```
tmpdir / tmpprefix .in  
tmpdir / tmpprefix .out  
tmpdir / tmpprefix .err
```

em que:

tmpdir é o valor especificado no parâmetro **TMPDIR**.

tmpprefix é o valor especificado no parâmetro **TMPPFILE**.

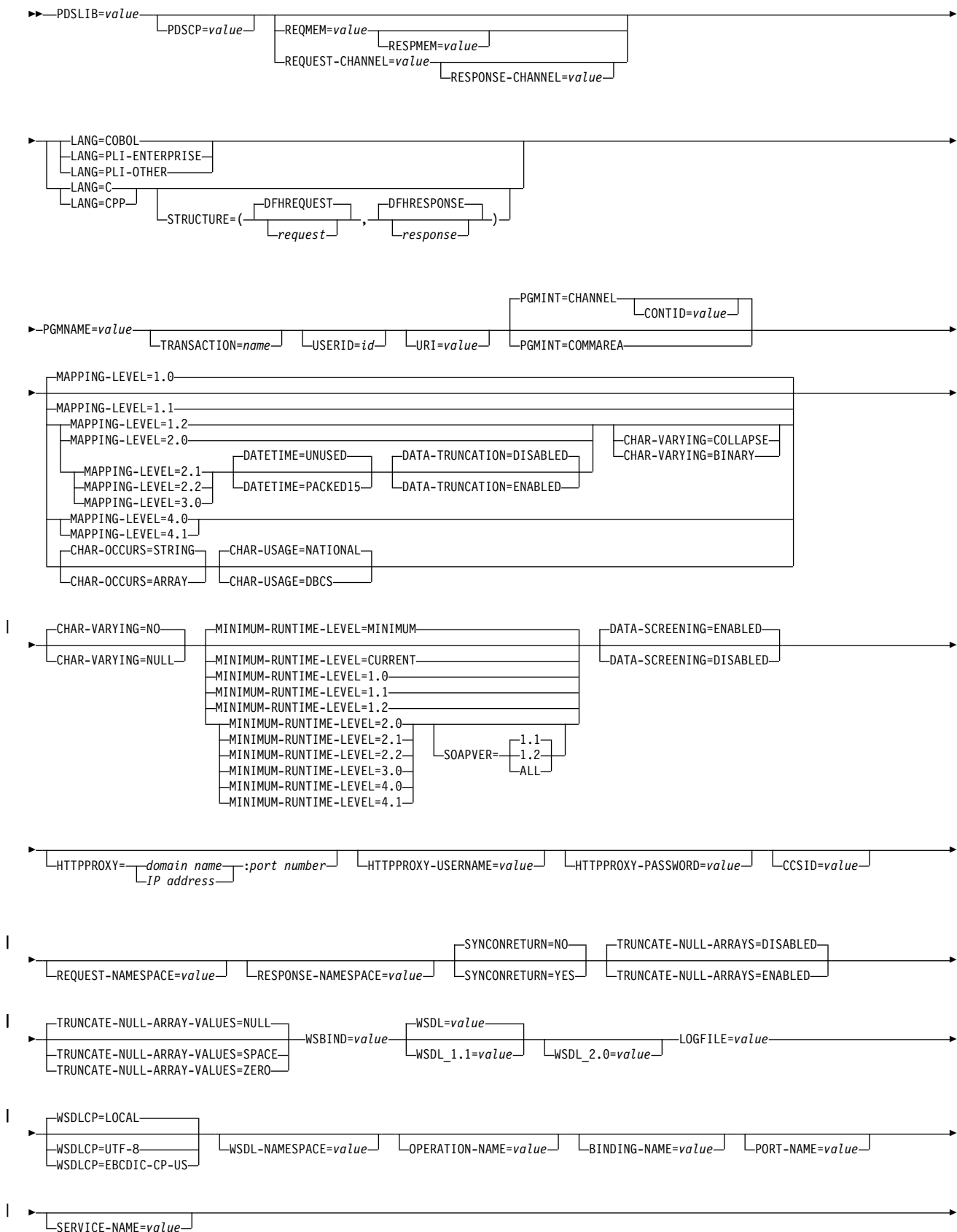
Os nomes padrão para os arquivos quando **TMPDIR** e **TMPPFILE** não são especificados, são conforme a seguir:

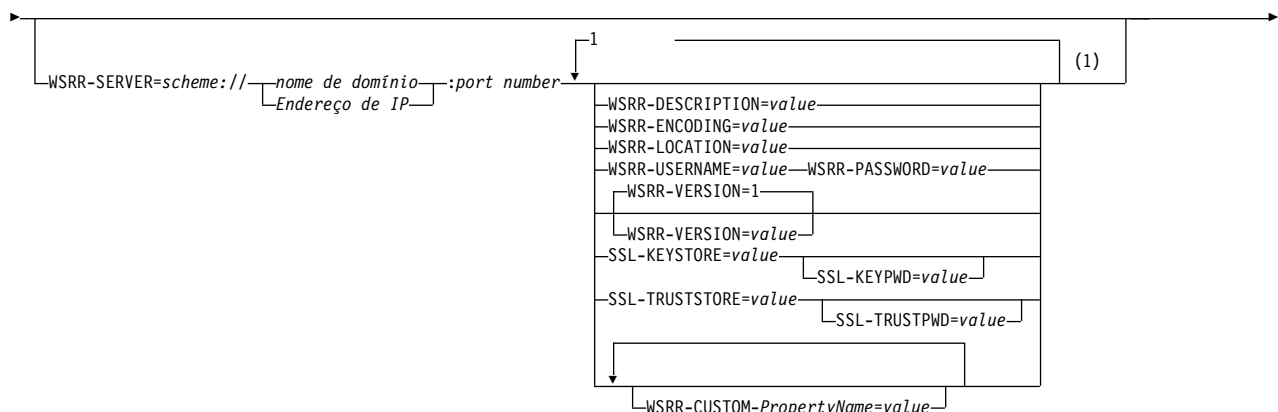
```
/tmp/LS2WS.in  
/tmp/LS2WS.out  
/tmp/LS2WS.err
```

Importante: O DFHLS2WS não bloqueia o acesso aos arquivos do z/OS UNIX ou aos membros do conjunto de dados. Portanto, se duas ou mais instâncias de DFHLS2WS forem executadas simultaneamente e usarem os mesmos arquivos da área de trabalho provisória, nada impede que uma tarefa sobrescreva os arquivos da área de trabalho enquanto outra tarefa está usando-os, o que pode causar falhas imprevisíveis.

Portanto, você será avisado para planejar uma convenção de nomenclatura e procedimentos operacionais que evitam essa situação. Por exemplo, é possível usar o parâmetro simbólico do sistema **SYSUID** para gerar nomes de arquivos de área de trabalho que são exclusivos para um usuário individual. Estes arquivos temporários são excluídos antes do encerramento da tarefa.

Parâmetros de Entrada para DFHLS2WS





Notas:

- 1 Cada um dos parâmetros WSRR que podem ser especificados quando o parâmetro **WSRR-SERVER** é configurado pode ser especificado somente uma vez. A exceção a essa regra é o parâmetro **WSRR-CUSTOM**, que você pode especificar, no máximo, 255 vezes.

Utilização de Parâmetros

- Você pode especificar os parâmetros de entrada em qualquer ordem.
- Cada parâmetro deve ser iniciado em uma nova linha.
- Um parâmetro e seu caractere de continuação, se você usar um, não devem se estender além da coluna 72; as colunas 73 a 80 devem conter espaços em branco.
- Se um parâmetro é muito longo para se ajustar em uma única linha, use um caractere de asterisco (*) no final da linha para indicar que o parâmetro continua na próxima linha. Tudo, incluindo espaços, antes do asterisco é considerado parte do parâmetro. Por exemplo:

```
WSBIND=wsbinddir*
      /app1
```

é equivalente a

```
WSBIND=wsbinddir/app1
```

- Um caractere # na primeira posição de caractere da linha é um caractere de comentário. A linha é ignorada.
- Uma vírgula na posição de último caractere da linha é um separador de linha opcional e é ignorado.

Descrições de Parâmetros

BINDING-NAME = *value*

Especifica o nome de ligação que é usado no documento WSDL gerado. Se nenhum valor for fornecido, um nome de ligação padrão será gerado usando o valor do parâmetro **PGMNAME** seguido por "HTTPSoapBinding". Se SOAPVER for configurado como ALL, um sufixo "12" será anexado ao nome da ligação SOAP 1.2.

CCSID = *value*

Especifica o CCSID que é usado no tempo de execução para codificar dados de caractere na estrutura de dados do aplicativo. O valor desse parâmetro substitui o valor do parâmetro de inicialização do sistema **LOCALCCSID**. O *value* deve ser um CCSID EBCDIC que seja suportado pelos serviços de conversão Java e z/OS (consulte z/OS Unicode Services User's Guide and Reference). Se

you do not specify this parameter, the data structure of the application will be encoded using the CCSID specified in the initialization parameter of the system.

This parameter can be used with any level of mapping.

CHAR-VARYING = { **NO** | **NULL** | **COLLAPSE** | **BINARY** }

Specifies how character fields in the language structure are mapped when the mapping level is 1.2 or higher. A character field in COBOL is a *Picture* clause of type *X*, for example *PIC(X) 10*; a character field in C/C++ is a matrix of characters. You can select these options:

NO Character fields are mapped for an `<xsd:string>` and are processed as fixed-length fields. The maximum length of the data is equal to the length of the field. **NO** is the default value for the **CHAR-VARYING** parameter for COBOL and PL/I at mapping levels 2.0 and earlier.

This value does not apply to Enterprise language structures and other PL/I.

NULL Character fields are mapped for an `<xsd:string>` and are processed as chains with null termination. CICS includes a null character of termination when transforming a message from SOAP. The maximum length of the character chain is calculated as a character less than the length indicated in the language structure. **NULL** is the default value for the **CHAR-VARYING** parameter for C/C++.

This value does not apply to Enterprise language structures and other PL/I.

COLLAPSE

Character fields are mapped for an `<xsd:string>`. Spaces at the end of the field are not included in the SOAP message. The SOAP message input is analyzed to remove all spaces at the beginning, end, and within the message. **COLLAPSE** is the default value for the **CHAR-VARYING** parameter for COBOL and PL/I at mapping level 2.1 and later.

For more information about variable length and space values, see Support for variable length and space values.

BINARY

Character fields are mapped for an `<xsd:base64binary>` and are processed as fixed-length fields. The **BINARY** value in the **CHAR-VARYING** parameter is available only at mapping level 2.1 and later.

CHAR-OCCURS = { **STRING** | **ARRAY** }

Specifies how character matrices in the language structure are mapped when the mapping level is 4.0 or higher. For example, *PIC X OCCURS 20*. This parameter is for use only with COBOL.

ARRAY

Character matrices are mapped for an XML matrix. This means that each character is mapped as an individual XML element. This is also the behavior at mapping levels 3.0 and earlier.

CADEIA

As matrizes de caracteres são mapeadas para uma sequência XML. Isso significa que a matriz COBOL inteira é mapeada como um elemento XML único.

CHAR-USAGE = { NATIONAL | DBCS }

Em COBOL, o tipo de dado nacional, PIC N, pode ser usado para dados UTF-16 ou DBCS. Essa configuração é controlada pela opção do compilador NSYMBOL. Deve-se configurar o parâmetro **CHAR-USAGE** no assistente com o mesmo valor que a opção do compilador NSYMBOL para assegurar que os dados sejam manipulados apropriadamente. Isso geralmente é configurado como CHAR-USAGE=NATIONAL quando você usa UTF-16.

DBCS Dados dos campos PIC (n) são tratados como dados codificados em DBCS.

NATIONAL

Dados dos campos PIC (n) são tratados como dados codificados em UTF-16.

CONTID = *value*

Em um provedor de serviços, especifica o nome do contêiner que contém a estrutura de dados de nível superior usada para representar uma mensagem SOAP.

O comprimento do contêiner que o CICS transmite para o programa de aplicativo de destino é o maior dos comprimentos do contêiner de solicitação e do contêiner de resposta.

DATA-SCREENING = { ENABLED | DISABLED }

Especifica se os dados fornecidos pelo aplicativo foram verificados com relação aos erros.

ENABLED

Quaisquer dados inconsistentes de tempo de execução fornecidos pelo aplicativo com a estrutura de linguagem são tratados como um erro e a mensagem DFHPI1010 é emitida. Uma resposta de erro é retornada ao aplicativo.

DISABLED

Os valores dos dados de tempo de execução fornecidos pelo aplicativo que são inconsistentes com a estrutura de linguagem são substituídos pelos valores padrão. Por exemplo, um zero substitui um valor inválido em um campo numérico. A mensagem DFHPI1010 não é emitida e uma resposta normal é retornada ao aplicativo. Esse recurso pode ser usado para evitar as respostas de erro INVALID_PACKED_DEC e INVALID_ZONED_DEC que são geradas a partir de campos de saída não inicializados.

DATA-TRUNCATION = { DISABLED | ENABLED }

Especifica se os dados de comprimento variável são tolerados em uma estrutura de campo de comprimento fixo:

DISABLED

Se os dados forem menores do que o comprimento fixo que o CICS está esperando, o CICS rejeitará os dados truncados e emitirá uma mensagem de erro.

ENABLED

Se os dados são menores do que o comprimento fixo que o CICS está esperando, o CICS tolera os dados truncados e processa os dados ausentes como valores nulos.

DATETIME = { **UNUSED** | **PACKED15** }

Especifica se campos ABSTIME em potencial na estrutura de linguagem de alto nível são mapeados como registros de data e hora:

PACKED15

Campos decimais compactados de comprimento 15 (8 bytes) são tratados como campos CICS ABSTIME e mapeados como registros.

UNUSED

Campos decimais compactados de comprimento 15 (8 bytes) não são tratados como registros de data e hora.

Você pode configurar este parâmetro no nível de mapeamento 3.0.

HTTPPROXY = { *domain name* : *port number* | *IP address* : *port number* }

Se seu WSDL contiver referências a outros arquivos WSDL que estão localizados na Internet e o sistema no qual você está executando o DFHLS2WS usar um servidor proxy para acessar a Internet, especifique o nome de domínio ou o endereço IP e o número da porta do servidor proxy. Por exemplo:

HTTPPROXY=proxy.example.com:8080

Em outros casos, este parâmetro não é necessário.

HTTPPROXY-PASSWORD = *value*

Especifica a senha do proxy HTTP que deve ser usada com **HTTPPROXY-USERNAME** se o sistema no qual você está executando o DFHLS2WS utiliza um servidor proxy HTTP para acessar a Internet e o servidor proxy HTTP utiliza autenticação básica. É possível usar esse parâmetro somente quando você também especifica **HTTPPROXY**.

HTTPPROXY-USERNAME = *value*

Especifica o nome de usuário do proxy HTTP que deve ser usado com **HTTPPROXY-PASSWORD** se o sistema no qual você está executando o DFHLS2WS utiliza um servidor proxy HTTP para acessar a Internet e o servidor proxy HTTP utiliza autenticação básica. É possível usar esse parâmetro somente quando você também especifica **HTTPPROXY**.

LANG = **COBOL**

Especifica que a linguagem de programação da estrutura de linguagem de alto nível é COBOL.

LANG = **PLI-ENTERPRISE**

Especifica que a linguagem de programação da estrutura de linguagem de alto nível é Enterprise PL/I.

LANG = **PLI-OTHER**

Especifica que a linguagem de programação da estrutura de linguagem de alto nível é um nível de PL/I diferente de Enterprise PL/I.

LANG = **C**

Especifica que a linguagem de programação da estrutura de linguagem de alto nível é C.

LANG = **CPP**

Especifica que a linguagem de programação da estrutura de linguagem de alto nível é C++.

LOGFILE = *value*

O nome completo do arquivo z/OS UNIX no qual o DFHLS2WS grava seu log de atividades e informações de rastreamento. DFHLS2WS cria o arquivo, mas não a estrutura de diretório, se ela ainda não existir.

Normalmente, você não usa esse arquivo, mas ele poderá ser solicitado pela organização de serviço IBM caso você encontre problemas com o DFHLS2WS.

MAPPING-LEVEL = { 1.0 | 1.1 | 1.2 | 2.0 | 2.1 | 2.2 | 3.0 | 4.0 | 4.1 }

Especifica o nível de mapeamento que o DFHLS2WS usa ao gerar o arquivo de ligação de serviço da web e a descrição de serviços da web. Você pode selecionar estas opções:

- 1.0** Este nível de mapeamento é o padrão. Ele indica que o arquivo de ligação de serviço da web é gerado usando os níveis de mapeamento de CICS TS 3.1.
- 1.1** Use este mapeamento para gerar novamente um arquivo de ligação neste nível específico.
- 1.2** Neste nível de mapeamento, é possível usar o parâmetro **CHAR-VARYING** para controlar como as matrizes de caracteres são processadas no tempo de execução. As matrizes **VARYING** e **VARYINGZ** também são suportadas em PL/I.
- 2.0** Use este nível de mapeamento em uma região do CICS TS 3.2 ou posterior para tirar vantagem dos aprimoramentos no mapeamento entre a estrutura de linguagem e o arquivo de ligação de serviços da web.
- 2.1** Use este nível de mapeamento com uma região do CICS TS 3.2 que tenha o APAR PK59794 aplicado ou com qualquer região mais recente do que o CICS TS 3.2. Neste nível de mapeamento é possível tirar vantagem dos novos valores para o parâmetro **CHAR-VARYING**, **COLLAPSE** e **BINARY**. Os campos **FILLER** em COBOL e os campos ***** em PL/I são sistematicamente ignorados neste nível de mapeamento, os campos não aparecem no documento WSDL gerado e um intervalo apropriado é deixado nas estruturas de dados no tempo de execução.
- 2.2** Use este nível de mapeamento com uma região do CICS TS 3.2 que tenha o APAR PK69738 aplicado ou com qualquer região mais recente do CICS TS 3.2 para tirar vantagem de aprimoramentos de mapeamento ao usar DFHWS2LS.
- 3.0** Use este nível de mapeamento com uma região do CICS TS 4.1. Neste nível de mapeamento é possível criar um serviço da web a partir de um aplicativo que usa muitos contêineres em sua interface configurando os parâmetros **REQUEST-CHANNEL** e **RESPONSE-CHANNEL**. Também é possível mapear campos **dateTime** para os registros de data e hora XML configurando o parâmetro **DATETIME**.
- 4.0** Use este nível de mapeamento com uma região do CICS TS 5.2 ou mais recente. Neste nível de mapeamento é possível usar campos **OCCURS DEPENDING ON** de COBOL e o parâmetro **CHAR-OCCURS**.
- 4.1** Use este nível de mapeamento para o suporte da matriz truncável, com uma região CICS TS V5.2 que tem o APAR PI67641 aplicado ou uma região CICS TS V5.3 ou mais recente.

Para obter mais informações sobre níveis de mapeamento, consulte Níveis de mapeamento para o assistentes do 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 o ambiente de tempo de execução mínimo do CICS no qual o arquivo de ligação de serviço da web pode ser implementado. Se você selecionar um nível que não corresponde aos outros parâmetros que você especificou, você receberá uma mensagem de erro. Você pode selecionar estas opções:

MINIMUM

O menor nível de tempo de execução possível do CICS é alocado automaticamente de acordo com os parâmetros que você especificou.

- 1.0** O arquivo de ligação de serviço da web gerado é implementado com sucesso em uma região do CICS TS 3.1 que não tem as APARs PK15904 e PK23547 aplicadas. Alguns parâmetros não estão disponíveis neste nível de tempo de execução.
- 1.1** O arquivo de ligação de serviço da web gerado é implementado com sucesso em uma região do CICS TS 3.1 que tem pelo menos o APAR PK15904 aplicado. É possível usar um nível de mapeamento 1.1 ou anterior para o parâmetro **MAPPING-LEVEL**. Alguns parâmetros não estão disponíveis neste nível de tempo de execução.
- 1.2** O arquivo de ligação de serviço da web gerado é implementado com sucesso em uma região do CICS TS 3.1 que tem ambos os APARs, PK15904 e PK23547, aplicados. É possível usar um nível de mapeamento 1.2 ou anterior para o parâmetro **MAPPING-LEVEL**. Alguns parâmetros não estão disponíveis neste nível de tempo de execução.
- 2.0** O arquivo de ligação de serviço da web gerado é implementado com sucesso em um CICS TS 3.2 ou em uma região posterior. É possível usar um nível de mapeamento de 2.0 ou anterior para o parâmetro **MAPPING-LEVEL**. Alguns parâmetros não estão disponíveis neste nível de tempo de execução.
- 2.1** O arquivo de ligação de serviço da web gerado é implementado com sucesso em uma região do CICS TS 3.2 que tem o APAR PK59794 aplicado ou em qualquer região posterior ao CICS TS 3.2. É possível usar um nível de mapeamento 2.1 ou anterior para o parâmetro **MAPPING-LEVEL**. Alguns parâmetros não estão disponíveis neste nível de tempo de execução.
- 2.2** O arquivo de ligação de serviço da web gerado é implementado com sucesso em uma região do CICS TS 3.2 que tem o APAR PK69738 aplicado ou em qualquer região posterior ao CICS TS 3.2. Com este nível de tempo de execução, é possível usar um nível de mapeamento 2.2 ou anterior para o parâmetro **MAPPING-LEVEL**. Alguns parâmetros não estão disponíveis neste nível de tempo de execução.
- 3,0** O arquivo de ligação de serviço da web gerado é implementado com sucesso em um CICS TS 4.1 ou em uma região posterior. Com este nível de tempo de execução, é possível usar um nível de mapeamento 3.0 ou anterior para o parâmetro **MAPPING-LEVEL**. Alguns parâmetros não estão disponíveis neste nível de tempo de execução.
- 4.0** O arquivo de ligação de serviço da web gerado é implementado com sucesso em um CICS TS 5.2 ou em uma região posterior. Com este nível de tempo de execução, é possível usar um nível de mapeamento

4.0 ou anterior para o parâmetro **MAPPING-LEVEL**. É possível utilizar qualquer parâmetro opcional neste nível.

- 4.1** O arquivo de ligação de serviços da web gerado é implementado com sucesso em uma região CICS V5.2 que possui o APAR PI67641 aplicado ou em qualquer região CICS V5.3 ou mais recente. Com este nível de tempo de execução, é possível utilizar um nível de mapeamento 4.1 ou anterior para o parâmetro **MAPPING-LEVEL**.

CURRENT

O arquivo de ligação de serviço da web gerado é implementado com sucesso em uma região CICS no mesmo nível de tempo de execução que você está usando para gerar o arquivo de ligação de serviço da web.

OPERATION-NAME = value

Especifica o nome da operação que é usado no documento WSDL gerado. Se nenhum valor for fornecido, um nome padrão será gerado usando o valor do parâmetro **PGMNAME** seguido pelo valor **operation**.

PDSLIB = value

Especifica o nome do conjunto de dados particionados que contém as estruturas de dados de linguagem de alto nível a serem processadas. Os membros do conjunto de dados usados para a solicitação e a resposta são especificados nos parâmetros **REQMEM** e **RESPMEM** respectivamente.

Restrição: Os registros no conjunto de dados particionados deve ter um comprimento fixo de 80 bytes.

PDSCP = value

Especifica a página de códigos usada nos membros do conjunto de dados particionados especificados nos parâmetros **REQMEM** e **RESPMEM**, em que *value* é um número de CCSID ou um número da página de código Java. Se este parâmetro não for especificado, a página de códigos do z/OS UNIX System Services será usada. Por exemplo, você pode especificar **PDSCP = 037**.

PGMINT = { CHANNEL | COMMAREA }

Para um provedor de serviços, especifica como o CICS transmite dados ao programa de aplicativo de destino:

CHANNEL

CICS usa uma interface do canal para transmitir dados ao programa de aplicativo de destino.

- Em níveis de mapeamento anteriores a 3.0, o canal pode conter somente um contêiner, que é usado para a entrada e a saída. Use o parâmetro **CONTID** para especificar o nome do contêiner. O nome padrão é DFHWS-DATA.
- No nível de mapeamento 3.0, o canal pode conter vários contêineres. Use os parâmetros **REQUEST-CHANNEL** e **RESPONSE-CHANNEL**. Não especifique **PDSLIB**, **REQMEM** ou **RESPMEM**.

COMMAREA

O CICS usa uma área de comunicação (COMMAREA) para transmitir dados para o programa de aplicativo de destino.

Quando o programa de aplicativo de destino tiver processado a solicitação, ele deverá usar o mesmo mecanismo para retornar a resposta. Se a solicitação foi recebida em uma área de comunicação, a resposta deverá ser retornada na área de comunicação; se a solicitação foi recebida em um contêiner, a resposta deverá ser retornada em um contêiner. O comprimento da área de

comunicação ou contêiner que o CICS transmite para o programa de aplicativo de destino é o maior dos comprimentos da área ou do contêiner de comunicação da solicitação e da área ou contêiner de comunicação de resposta.

PGMNAME = *value*

Especifica o nome do recurso PROGRAM do CICS para o programa de aplicativo de destino que será exposto como um serviço da web. O suporte do serviço da web do CICS vinculará a este programa.

PORT-NAME = *value*

Especifica o nome que é usado para a porta e o portType no documento WSDL gerado. Se nenhum valor for fornecido, um nome padrão será gerado usando o valor do parâmetro **PGMNAME** seguido por "Port". Se SOAPVER estiver configurado como ALL, um sufixo "12" será anexado ao nome da porta SOAP 1.2.

REQMEM = *value*

Especifica o nome do membro do conjunto de dados particionados que contém a estrutura de linguagem de alto nível para a solicitação de serviço da web. Para um provedor de serviços, a solicitação de serviço da web é a entrada para o programa de aplicativo.

REQUEST-CHANNEL = *value*

Especifica o nome e o local de um documento de descrição de canal. A descrição de canal descreve os contêineres que o aplicativo do provedor de serviço da web pode usar em sua interface ao receber uma mensagem SOAP de um solicitante de serviço da web. A descrição de canal é um documento XML que devem estar em conformidade com o esquema de canal fornecido pelo CICS.

Você pode usar esse parâmetro somente no nível de mapeamento 3.0.

REQUEST-NAMESPACE = *value*

Especifica o namespace do esquema XML para a mensagem de solicitação na descrição do serviço da web gerado. Se você não especificar este parâmetro, o CICS gerará um namespace automaticamente.

RESPMEM = *value*

Especifica o nome do membro do conjunto de dados particionados que contém a estrutura de linguagem de alto nível para a resposta de serviço da web. Para um provedor de serviços, a resposta de serviço da web é a saída do programa de aplicativo.

Omita esse parâmetro se nenhuma resposta estiver envolvida; ou seja, para mensagens unidirecionais.

RESPONSE-CHANNEL = *value*

Especifica o nome e o local de um documento de descrição de canal. A descrição de canal descreve os contêineres que o aplicativo do provedor de serviços da web pode usar em sua interface ao enviar uma mensagem de resposta SOAP para um solicitante de serviço da web. A descrição de canal é um documento XML que deve estar em conformidade com o esquema de canal fornecido pelo CICS.

Você pode usar esse parâmetro somente no nível de mapeamento 3.0.

RESPONSE-NAMESPACE = *value*

Especifica o namespace do esquema XML para a mensagem de resposta na descrição do serviço da web gerado. Se você não especificar este parâmetro, o CICS gerará um namespace automaticamente.

SERVICE-NAME = *value*

Especifica o nome do serviço que é usado no documento WSDL gerado. Se nenhum valor for fornecido, um nome de serviço padrão será gerado usando o valor do parâmetro **PGMNAME** seguido por "Service".

SOAPVER = { 1.1 | 1.2 | ALL }

Especifica o nível de SOAP a ser usado na descrição do serviço da web gerado. Este parâmetro está disponível somente quando o **MINIMUM-RUNTIME-LEVEL** é configurado como 2.0 ou superior.

1.1 O protocolo SOAP 1.1 é usado como a ligação para a descrição de serviço da web.

1.2 O protocolo SOAP 1.2 é usado como a ligação para a descrição de serviços da web.

ALL O protocolo SOAP 1.1 ou 1.2 pode ser usado como a ligação para a descrição de serviços da web.

Se você não especificar um valor para esse parâmetro, o valor padrão dependerá da versão do WSDL que você deseja criar:

- Se você requerer somente WSDL 1.1, a ligação SOAP 1.1 será utilizada.
- Se você requerer somente WSDL 2.0, a ligação SOAP 1.2 será utilizada.
- Se você requerer WSDL 1.1 e WSDL 2.0, ambas as ligações, SOAP 1.1 e 1.2, serão usadas para cada descrição de serviços da web.

SSL-KEYSTORE = *value*

Este parâmetro opcional especifica o local completo do arquivo de armazenamento de chaves.

Use este parâmetro se desejar que o assistente de serviços da web use a criptografia secure sockets layer (SSL) para se comunicar em uma rede com um IBM WebSphere Service Registry and Repository (WSRR).

SSL-KEYPWD = *value*

Este parâmetro opcional especifica a senha para o armazenamento de chaves.

Use este parâmetro se desejar que o assistente de serviços da web use a criptografia secure sockets layer (SSL) para se comunicar em uma rede com um IBM WebSphere Service Registry and Repository (WSRR).

SSL-TRUSTSTORE = *value*

Este parâmetro opcional especifica o local completo do arquivo de armazenamento confiável.

Use este parâmetro se desejar que o assistente de serviços da web use a criptografia secure sockets layer (SSL) para se comunicar em uma rede com um IBM WebSphere Service Registry and Repository (WSRR).

SSL-TRUSTPWD = *value*

Este parâmetro opcional especifica a senha para o armazenamento confiável.

Use este parâmetro se desejar que o assistente de serviços da web use a criptografia secure sockets layer (SSL) para se comunicar em uma rede com um IBM WebSphere Service Registry and Repository (WSRR).

STRUCTURE = (*request* , *response*)

Apenas para C e C++, especifica os nomes das estruturas de alto nível contidas nos membros do conjunto de dados particionados que são especificados nos parâmetros **REQMEM** e **RESPMEM**:

request

Especifica o nome da estrutura de alto nível que contém a solicitação quando o parâmetro **REQMEM** é especificado. O valor padrão é DFHREQUEST.

O membro do conjunto de dados particionados deve conter uma estrutura de alto nível com o nome que você especificar ou uma estrutura denominada DFHREQUEST se você não especificar um nome.

response

Especifica o nome da estrutura de alto nível que contém a resposta quando o parâmetro **RESPMEM** é especificado. O valor padrão é DFHRESPONSE.

Se você especificar um valor, o membro do conjunto de dados particionados deverá conter uma estrutura de alto nível com o nome que você especificar ou uma estrutura denominada DFHRESPONSE se você não especificar um nome.

SYNCONRETURN = { **NO** | **YES** }

Especifica se o serviço da web remoto pode emitir um ponto de sincronização.

NO O serviço da web remoto não pode emitir um ponto de sincronização. Este valor é o padrão. Se o serviço da web remoto emitir um ponto de sincronização, ele falhará com um encerramento anormal de ADPL.

YES O serviço da web remoto pode emitir um ponto de sincronização. Se você selecionar YES, a tarefa remota será confirmada como uma unidade de trabalho separada quando o controle retornar do serviço da web remoto. Se o serviço da web remoto atualizar um recurso recuperável e ocorrer uma falha após ele retornar, a atualização para esse recurso não poderá ser restaurada.

TRANSACTION = *name*

Em um provedor de serviços, este parâmetro especifica o nome com 1 a 4 caracteres de uma transação de alias que pode iniciar o pipeline. O valor desse parâmetro é usado para definir o atributo TRANSACTION do recurso URIMAP quando ele é criado automaticamente usando o comando de varredura **PIPELINE**.

Caracteres aceitáveis:

A-Z a-z 0-9 \$ @ # _ < >

TRUNCATE-NULL-ARRAYS = { **DISABLED** | **ENABLED** }

Especifica como matrizes estruturadas são processadas no nível de mapeamento 4.1 ou superior. Se ativado, o CICS tentará reconhecer registros vazios em uma matriz (consulte TRUNCATE-NULL-ARRAY-VALUES para obter mais informações sobre como identificar os registros vazios). Se cinco registros de matriz vazia consecutivos forem detectados, a matriz será truncada no primeiro registro ao gerar XML/JSON. Esse recurso de truncamento é ativado somente para matrizes com conteúdo estruturado, matrizes de campos primitivos simples não estão sujeitas a truncamento. O truncamento de matrizes pode resultar em uma representação mais concisa dos dados em JSON/XML, mas não é sem risco. Se cinco registros de dados consecutivos forem identificados incorretamente como armazenamento não inicializado (talvez porque eles legitimamente contêm valores baixos), poderá haver perda de dados. Se TRUNCATE-NULL-ARRAYS estiver ativado e TRUNCATE-NULL-ARRAY-VALUES não estiver configurado, o valor padrão para TRUNCATE-NULL-ARRAY-VALUES será usado.

TRUNCATE-NULL-ARRAY-VALUES = { NULL | **SPACE** | **ZERO** }

Especifica quais valores são tratados como vazios para processamento de TRUNCATE-NULL-ARRAYS no nível de mapeamento 4.1 ou superior. Por padrão, o valor nulo (0x00 ou valores baixos) é tratado como vazio. Se todos os bytes de armazenamento em um registro de uma matriz estruturada contêm valores nulos, o registro inteiro é considerado vazio. Um ou mais dos valores NULL, SPACE e ZERO podem ser especificados em uma lista separada por vírgula, em que NULL indica um caractere nulo (0x00), SPACE indica um espaço SBCS EBCDIC (0x40) e ZERO indica um zero decimal zonado não sinalizado (0xF0). Qualquer combinação correspondente dos bytes selecionados em um registro de matriz estruturada fará com que o registro inteiro seja identificado como vazio. Se TRUNCATE-NULL-ARRAY-VALUES tem um valor definido, TRUNCATE-NULL-ARRAYS deve ser ativado.

URI = *value*

Este parâmetro especifica o URI relativo ou absoluto que um cliente utilizará para acessar o serviço da web. O CICS usa o valor especificado quando ele gera um recurso URIMAP a partir do arquivo de ligação de serviço da web criado por DFHLS2WS. O parâmetro especifica o componente de caminho do URI ao qual a definição de URIMAP se aplica.

USERID = *id*

Em um provedor de serviços, este parâmetro especifica um ID do usuário com 1 a 8 caracteres, que pode ser usado por qualquer Web client. Para uma resposta gerada por aplicativo ou um serviço da web, a transação de alias é conectada com este ID do usuário. O valor desse parâmetro é usado para definir o atributo USERID do recurso URIMAP quando ele é criado automaticamente usando o comando de varredura **PIPELINE**.

Caracteres aceitáveis:

A-Z a-z 0-9 \$ @ #

WSBIND = *value*

O nome completo do z/OS UNIX do arquivo de ligação de serviço da web. DFHLS2WS cria o arquivo, mas não a estrutura de diretório, se ela ainda não existir. A extensão do arquivo é .wsbind.

WSDL = *value*

O nome completo do z/OS UNIX do arquivo no qual a descrição de serviço da web é gravada. A descrição do serviço da web está em conformidade com a especificação WSDL 1.1. DFHLS2WS cria o arquivo, mas não a estrutura de diretório, se ela ainda não existir. A extensão do arquivo é .wsdl.

WSDL_1.1 = *value*

O nome completo do z/OS UNIX do arquivo no qual a descrição de serviço da web é gravada. A descrição do serviço da web está em conformidade com a especificação WSDL 1.1. DFHLS2WS cria o arquivo, mas não a estrutura de diretório, se ela ainda não existir. A extensão do arquivo é .wsdl. Este parâmetro produz o mesmo resultado que o parâmetro **WSDL**, portanto, você pode especificar somente um ou outro.

WSDL_2.0 = *value*

O nome completo do z/OS UNIX do arquivo no qual a descrição de serviço da web é gravada. A descrição do serviço da web em conformidade com a especificação WSDL 2.0. DFHLS2WS cria o arquivo, mas não a estrutura de diretório, se ela ainda não existir. A extensão do arquivo é .wsdl. Este

parâmetro pode ser usado com os parâmetros **WSDL** ou **WSDL_1.1**. Ele está disponível somente quando o **MINIMUM-RUNTIME-LEVEL** é configurado como 2.0 ou superior.

WSDLCP = { **LOCAL** | **UTF-8** | **EBCDIC-CP-US** }

Especifica a página de códigos que é utilizada para gerar o documento WSDL.

LOCAL

Especifica que o documento WSDL é gerado utilizando a página de códigos local e nenhuma tag de codificação é gerada no documento WSDL.

UTF-8 Especifica que o documento WSDL é gerado utilizando a página de códigos UTF-8. Uma tag de codificação é gerada no documento WSDL. Se você especificar esta opção, deverá assegurar que a codificação permaneça correta ao copiar o documento WSDL entre diferentes plataformas.

EBCDIC-CP-US

Esse valor especifica que o documento WSDL é gerado utilizando a página de códigos US EBCDIC. Uma tag de codificação é gerada no documento WSDL.

WSDL-NAMESPACE = *value*

Especifica o namespace para CICS usar no documento WSDL gerado.

Se você não especificar este parâmetro, o CICS gerará um namespace automaticamente.

WSRR-CUSTOM- *PropertyName* = *value*

Use este parâmetro opcional para incluir metadados customizados no documento WSDL no WSRR. Os pares **WSRR-CUSTOM- *PropertyName*** = *value* são incluídos no documento WSDL e aparecem no WSRR sem o prefixo **WSRR-CUSTOM**.

É possível especificar um máximo de 255 pares *PropertyName* = *value* customizados. Evite pares *PropertyName* = *value* duplicados e em branco.

Use esse parâmetro somente quando o parâmetro **WSRR-SERVER** for especificado.

WSRR-DESCRIPTION = *value*

Use este parâmetro opcional para especificar os metadados que descrevem o documento WSDL que está sendo publicado.

Use esse parâmetro somente quando o parâmetro **WSRR-SERVER** for especificado.

WSRR-ENCODING = *value*

Use este parâmetro opcional para especificar a codificação do conjunto de caracteres do documento WSDL. Se o parâmetro **WSRR-ENCODING** não for especificado, o WSRR usará o valor especificado no documento WSDL.

Use esse parâmetro somente quando o parâmetro **WSRR-SERVER** for especificado.

WSRR-LOCATION = *value*

Use este parâmetro opcional para especificar o URI que identifica o local do documento WSDL. Se este parâmetro não for especificado, o URI será padronizado com o nome do arquivo especificado no parâmetro **WSDL**. Por exemplo, se o valor do parâmetro **WSDL** for `wsrr/example.wsdl`, o valor do parâmetro **WSRR-LOCATION** será padronizado como `example.wsdl`.

Use esse parâmetro somente quando o parâmetro **WSRR-SERVER** for especificado.

WSRR-PASSWORD = *value*

Use este parâmetro opcional se você deve inserir uma senha para acessar o WSRR.

Se o parâmetro **WSRR-USERNAME** for especificado, você também deverá especificar esse parâmetro.

Use esse parâmetro somente quando o parâmetro **WSRR-SERVER** for especificado.

WSRR-SERVER = { *domain name* : *port number* | *IP address* : *port number* }

Use este parâmetro para especificar o local do servidor IBM WebSphere Service Registry and Repository (WSRR). Se este parâmetro for especificado, a validação de parâmetro WSRR será usada.

WSRR-USERNAME = *value*

Use este parâmetro opcional se for necessário especificar um nome de usuário para acessar o WSRR. Esse nome de usuário é usado pelo WSRR para configurar a propriedade de proprietário.

Use esse parâmetro somente quando o parâmetro **WSRR-SERVER** for especificado.

WSRR-VERSION = { 1 | *value* }

Use este parâmetro para configurar a propriedade da versão do documento WSDL no WSRR.

Use esse parâmetro somente quando o parâmetro **WSRR-SERVER** for especificado.

Outras informações

- O ID do usuário sob o qual o DFHLS2SC é executado deve ser configurado para usar o UNIX System Services. O ID do usuário deve ter permissão de leitura para a estrutura de arquivo e bibliotecas PDS do CICS z/OS UNIX e permissão de gravação para os diretórios especificados nos parâmetros **LOGFILE**, **WSBIND** e **WSDL**.
- O ID do usuário deve ter uma alocação de armazenamento suficientemente grande para executar Java.
- A JCL possui um comprimento de parâmetro máximo de 100 caracteres. Isso pode ser aumentado usando a instrução **STDPARM**, para obter mais informações, consulte *z/OS UNIX System Services User Guide*.

Exemplo

```
//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: WSDL para Conversão de Linguagem de Alto Nível

O procedimento DFHWS2LS gera uma estrutura de dados de linguagem de alto nível e um arquivo de ligação de serviço da web a partir de uma descrição de serviços da web. É possível usar DFHWS2LS ao expor um programa de aplicativo CICS como um provedor de serviços ou quando você constrói um solicitante de serviço.

Instruções de Controle da Tarefa para DFHWS2LS

JOB Inicia a tarefa.

EXEC Especifica o nome do procedimento (DFHWS2LS).

INPUT.SYSUT1 DD

Especifica a entrada. Os parâmetros de entrada geralmente são especificados no fluxo de entrada. No entanto, eles podem ser definidos em um conjunto de dados ou em um membro de um conjunto de dados particionados.

Parâmetros simbólicos

Os parâmetros simbólicos a seguir são definidos no DFHWS2LS:

JAVADIR = *path*

Especifica o nome do diretório Java que é usado pelo DFHWS2LS. O valor desse parâmetro é anexado a /usr/lpp/ para produzir um nome de caminho completo de /usr/lpp/*path*.

Normalmente, você não especifica este parâmetro. O valor padrão é o valor que foi fornecido à tarefa de instalação do CICS (DFHISTAR) no parâmetro **JAVADIR**.

PATHPREFIX = *prefix*

Especifica um prefixo opcional que estende o caminho do diretório z/OS UNIX usado em outros parâmetros. O padrão é a cadeia vazia.

Normalmente, você não especifica este parâmetro. O valor padrão é o valor que foi fornecido para a tarefa de instalação do CICS (DFHISTAR) no parâmetro **PATHPREFIX**.

TMPDIR = *tmpdir*

Especifica o local de um diretório no z/OS UNIX que o DFHWS2LS usa como uma área de trabalho provisória. O ID do usuário sob o qual a tarefa é executada deve ter permissão de leitura e gravação para esse diretório.

O valor padrão é /tmp.

TMPFILE = *tmpprefix*

Especifica um prefixo que o DFHWS2LS usa para construir os nomes dos arquivos de área de trabalho provisória.

O valor padrão é WS2LS.

USSDIR = *path*

Especifica o nome do diretório do CICS TS no sistema de arquivos de serviços do sistema UNIX. O valor desse parâmetro é anexado a /usr/lpp/cicsts/ para produzir um nome de caminho completo de /usr/lpp/cicsts/*path*.

Normalmente, você não especifica este parâmetro. O valor padrão é o valor que foi fornecido à tarefa de instalação do CICS (DFHISTAR) no parâmetro **USSDIR**.

SERVICE = *value*

Use este parâmetro somente quando orientado a fazer isso pelo Suporte IBM.

O Espaço de Trabalho Temporário

DFHWS2LS cria os três arquivos temporários a seguir no tempo de execução:

```
tmpdir / tmpprefix .in
tmpdir / tmpprefix .out
tmpdir / tmpprefix .err
```

em que:

tmpdir é o valor especificado no parâmetro **TMPDIR**.

tmpprefix é o valor especificado no parâmetro **TMPFILE**.

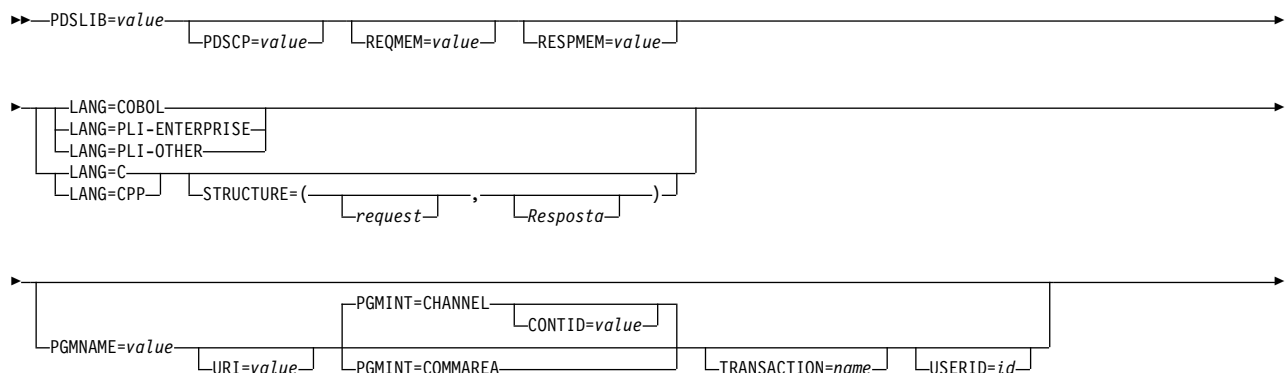
Os nomes padrão para os arquivos quando **TMPDIR** e **TMPFILE** não são especificados, são conforme a seguir:

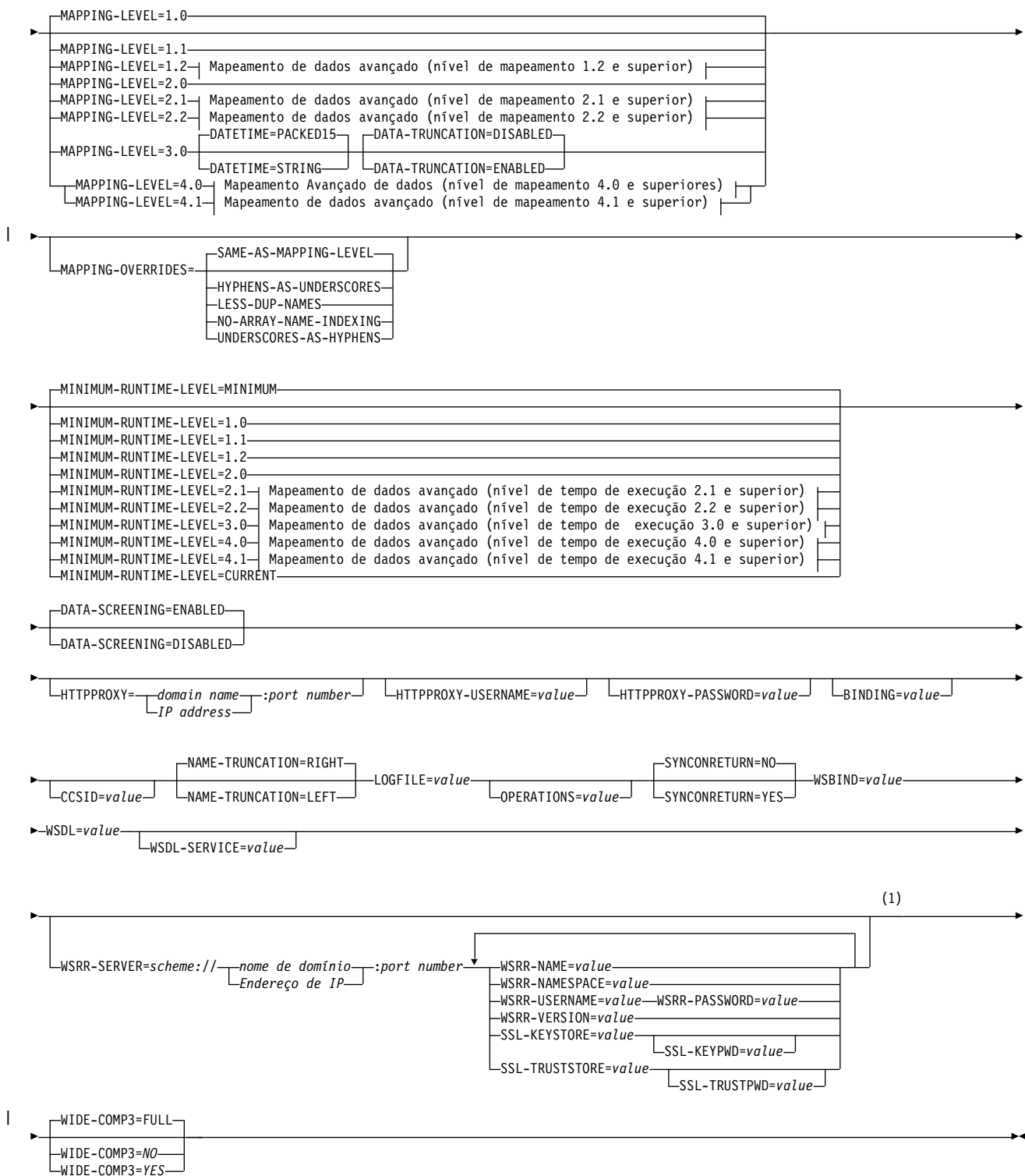
```
/tmp/WS2LS.in
/tmp/WS2LS.out
/tmp/WS2LS.err
```

Importante: O DFHWS2LS não bloqueia o acesso aos arquivos do z/OS UNIX ou aos membros do conjunto de dados. Portanto, se duas ou mais instâncias de DFHWS2LS forem executadas simultaneamente e usarem os mesmos arquivos de área de trabalho provisória, nada impede que uma tarefa sobrescreva os arquivos da área de trabalho enquanto outra tarefa está usando-os, levando a falhas imprevisíveis.

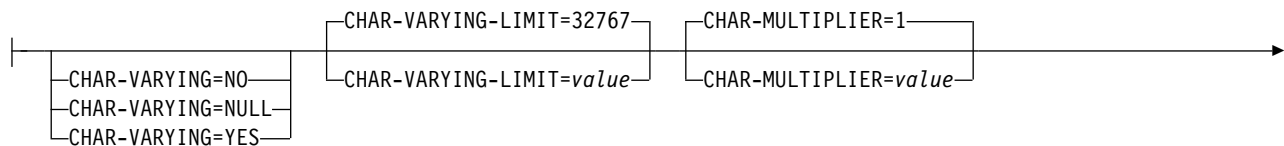
Portanto, você será avisado para planejar uma convenção de nomenclatura e procedimentos operacionais que evitam essa situação. Por exemplo, é possível usar o parâmetro simbólico do sistema **SYSUID** para gerar nomes de arquivos de área de trabalho que são exclusivos para um usuário individual. Estes arquivos temporários são excluídos antes do encerramento da tarefa.

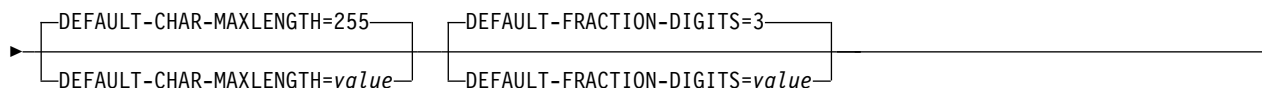
Parâmetros de Entrada para DFHWS2LS



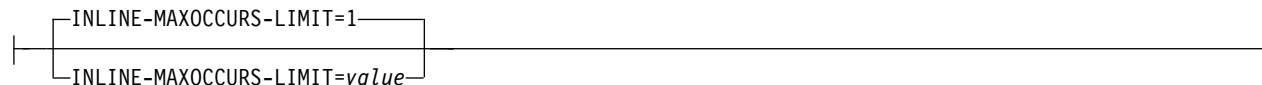


Mapeamento de Dados Avançado (Nível de Mapeamento 1.2 e Superior):

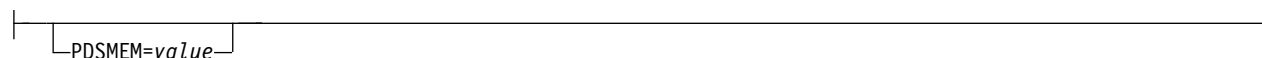




Mapeamento de Dados Avançado (Nível de Mapeamento 2.1 e Superior):



Mapeamento de Dados Avançado (nível de mapeamento 2.2 e superior):



Mapeamento de Dados Avançado (Nível de Tempo de Execução 2.1 e Superior):



Mapeamento de Dados Avançado (Nível de Tempo de Execução 3.0 e Superior):



Notas:

- 1 Cada um dos parâmetros WSRR que podem ser especificados quando o parâmetro **WSRR-SERVER** é configurado pode ser especificado somente uma vez.

Utilização de Parâmetros

- Você pode especificar os parâmetros de entrada em qualquer ordem.
- Cada parâmetro deve ser iniciado em uma nova linha.
- Um parâmetro e seu caractere de continuação, se você usar um, não devem se estender além da coluna 72; as colunas 73 a 80 devem conter espaços em branco.
- Se um parâmetro é muito longo para se ajustar em uma única linha, use um caractere de asterisco (*) no final da linha para indicar que o parâmetro continua na próxima linha. Tudo, incluindo espaços, antes do asterisco é considerado parte do parâmetro. Por exemplo:

```
WSBIND=wsbinddir*
      /app1
```

é equivalente a

```
WSBIND=wsbinddir/app1
```

- Um caractere # na primeira posição de caractere da linha é um caractere de comentário. A linha é ignorada.
- Uma vírgula na posição de último caractere da linha é um separador de linha opcional e é ignorado.

Descrições de Parâmetros

BINDING = *value*

Se a descrição de serviços da web contém mais de um elemento <wsdl:Binding>, use este parâmetro para especificar qual deve ser usado para gerar a estrutura de linguagem e o arquivo de ligação de serviço da web. Especifique o valor do atributo name que é usado no elemento <wsdl:Binding> na descrição de serviços da web.

CCSID = *value*

Especifica o CCSID que é usado no tempo de execução para codificar dados de caractere na estrutura de dados do aplicativo. O valor desse parâmetro substitui o valor do parâmetro de inicialização do sistema **LOCALCCSID**. O *value* deve ser um CCSID EBCDIC que seja suportado pelos serviços de conversão Java e z/OS (consulte z/OS Unicode Services User's Guide and Reference). Se você não especificar esse parâmetro, a estrutura de dados do aplicativo será codificada usando o CCSID especificado no parâmetro de inicialização do sistema.

Este parâmetro pode ser utilizado com qualquer nível de mapeamento.

CHAR-MULTIPLIER = { 1 | *value* }

Especifica o número de bytes a ser permitido para cada caractere quando o nível de mapeamento é 1.2 ou mais recente. O *value* deste parâmetro pode ser um número inteiro positivo no intervalo de 1 a 2.147.483.647. Todos os mapeamentos baseados em caracteres não numéricos estão sujeitos a este multiplicador. Campos binários, numéricos, zoneados e decimais compactados não estão sujeitos a este multiplicador.

Este parâmetro pode ser útil se, por exemplo, você está planejando usar caracteres DBCS onde pode optar por um multiplicador de 3 para permitir espaço para possíveis caracteres shift-out e shift-in em cada caractere de byte duplo no tempo de execução.

Quando você configura **CCSID=1200** (indicando UTF-16), os únicos valores válidos para **CHAR-MULTIPLIER** são 2 ou 4. Ao usar UTF-16, o valor padrão é 2. Use **CHAR-MULTIPLIER=2** quando esperar que os dados do aplicativo contenham caracteres que requerem 1 unidade de codificação UTF-16. Use **CHAR-MULTIPLIER=4** quando esperar que os dados do aplicativo contenham caracteres que requerem 2 unidades de codificação UTF-16.

Nota: A configuração de **CHAR-MULTIPLIER** como 1 não impede o uso de caracteres DBCS e sua configuração como 2 não impede o uso de pares substitutos UTF-16. No entanto, se caracteres largos forem usados rotineiramente, alguns valores válidos não se ajustarão no campo alocado. Se um valor **CHAR-MULTIPLIER** maior for usado, poderá ser possível armazenar mais caracteres no campo alocado que são válidos no XML. Deve-se tomar cuidado com relação à conformidade com as restrições de intervalo apropriadas.

CHAR-VARYING = { **NO** | **NULL** | **YES** }

Especifica como os dados de caracteres de comprimento variável são mapeados quando o nível de mapeamento é 1.2 ou superior. Os tipos de dados binários de comprimento variável são sempre mapeados para um contêiner ou uma estrutura variável. Se você não especificar esse parâmetro, o mapeamento padrão dependerá da linguagem especificada. Você pode selecionar estas opções:

NO Os dados de caractere de comprimento variável são mapeados como sequências de comprimento fixo.

NULL Os dados de caractere de comprimento variável são mapeados para sequências com terminação nula.

YES Dados de caractere de comprimento variável são mapeados para um tipo de dados CHAR VARYING em PL/I. Nas linguagens COBOL, C e C++, os dados de caracteres de comprimento variável são mapeados para uma representação equivalente que consiste em dois elementos relacionados: dados de comprimento e dados.

CHAR-VARYING-LIMIT = { 32767 | *value* }

Especifica o tamanho máximo de dados binários e de dados de caractere de comprimento variável que são mapeados para a estrutura de linguagem quando o nível de mapeamento é 1.2 ou superior. Se os dados de caractere ou binários forem maiores do que o valor especificado neste parâmetro, eles serão mapeados para um contêiner e o nome do contêiner será usado na estrutura de linguagem gerada. O valor pode variar de 0 até o padrão de 32.767 bytes.

CONTID = *value*

Em um provedor de serviços, especifica o nome do contêiner que contém a estrutura de dados de nível superior usada para representar uma mensagem SOAP.

O comprimento do contêiner que o CICS transmite para o programa de aplicativo de destino é o maior dos comprimentos do contêiner de solicitação e do contêiner de resposta.

DATA-SCREENING = { ENABLED | **DISABLED** }

Especifica se os dados fornecidos pelo aplicativo foram verificados com relação aos erros.

ENABLED

Quaisquer dados inconsistentes de tempo de execução fornecidos pelo aplicativo com a estrutura de linguagem são tratados como um erro e a mensagem DFHPI1010 é emitida. Uma resposta de erro é retornada ao aplicativo.

DISABLED

Os valores dos dados de tempo de execução fornecidos pelo aplicativo que são inconsistentes com a estrutura de linguagem são substituídos pelos valores padrão. Por exemplo, um zero substitui um valor inválido em um campo numérico. A mensagem DFHPI1010 não é emitida e uma resposta normal é retornada ao aplicativo. Esse recurso pode ser usado para evitar as respostas de erro INVALID_PACKED_DEC e INVALID_ZONED_DEC que são geradas a partir de campos de saída não inicializados.

DATA-TRUNCATION = { DISABLED | **ENABLED** }

Especifica se os dados de comprimento variável são tolerados em uma estrutura de campo de comprimento fixo:

DISABLED

Se os dados forem menores do que o comprimento fixo que o CICS está esperando, o CICS rejeitará os dados truncados e emitirá uma mensagem de erro.

ENABLED

Se os dados são menores do que o comprimento fixo que o CICS está esperando, o CICS tolera os dados truncados e processa os dados ausentes como valores nulos.

DATETIME = { **PACKED15** | **STRING** }

Especifica como os elementos <xsd:dateTime> são mapeados para a estrutura de linguagem.

PACKED15

O padrão é que qualquer elemento <xsd:dateTime> é processado como um registro de data e hora e mapeado para o formato CICS ABSTIME.

CADEIA

O elemento <xsd:dateTime> é processado como texto.

DEFAULT-CHAR-MAXLENGTH = { **255** | *value* }

Especifica o comprimento da matriz padrão de dados de caractere em caracteres para mapeamentos em que nenhum comprimento esteja implícito no documento de descrição de serviço da web, quando o nível de mapeamento é 1.2 ou superior. O valor desse parâmetro pode ser um número inteiro positivo no intervalo de 1 a 2.147.483.647.

DEFAULT-FRACTION-DIGITS = { **3** | *value* }

Especifica o número padrão de dígitos fracionários para uso em um tipo de esquema decimal XML. O padrão é 3. Para COBOL, o intervalo válido é de 0 a 17 ou de 0 a 30 se o parâmetro **WIDE-COMP3** está sendo usado. Para C ou PLI, o intervalo válido é 0-30.

HTTPPROXY = { *domain name : port number* | *IP address : port number* }

Se seu WSDL contiver referências a outros arquivos WSDL que estão localizados na Internet e o sistema no qual você está executando o DFHWS2LS utilizar um servidor proxy para acessar a Internet, especifique o nome de domínio ou o endereço IP e o número da porta do servidor proxy. Por exemplo:

HTTPPROXY=proxy.example.com:8080

Em outros casos, este parâmetro não é necessário.

HTTPPROXY-PASSWORD = *value*

Especifica a senha do proxy HTTP que deve ser usada com **HTTPPROXY-USERNAME** se o sistema no qual você está executando o DFHWS2LS utiliza um servidor proxy HTTP para acessar a Internet e o servidor proxy HTTP utiliza autenticação básica. É possível usar esse parâmetro somente quando você também especifica **HTTPPROXY**.

HTTPPROXY-USERNAME = *value*

Especifica o nome de usuário do proxy HTTP que deve ser usado com **HTTPPROXY-PASSWORD** se o sistema no qual você está executando o DFHWS2LS utiliza um servidor proxy HTTP para acessar a Internet e o servidor proxy HTTP utiliza autenticação básica. É possível usar esse parâmetro somente quando você também especifica **HTTPPROXY**.

INLINE-MAXOCCURS-LIMIT = { **1** | *value* }

Especifica se o conteúdo de repetição de variável sequencial é ou não utilizado com base no atributo maxOccurs. O conteúdo de repetição variável que é mapeado sequencialmente é colocado no contêiner atual com o resto da estrutura de linguagem gerada. O conteúdo de repetição variável é armazenado em duas partes, como um contador que armazena o número de ocorrências dos dados e como uma matriz que armazena cada ocorrência dos dados. A alternativa de mapeamento para conteúdo variavelmente repetido é o mapeamento baseado em contêiner, que armazena o número de ocorrências dos dados e o nome do contêiner onde os dados são colocados. Armazenar os dados em um contêiner separado tem implicações de desempenho que podem tornar o mapeamento sequencial preferível.

O parâmetro **INLINE-MAXOCCURS-LIMIT** está disponível somente no nível de mapeamento 2.1 em diante. O valor de **INLINE-MAXOCCURS-LIMIT** pode ser um número inteiro positivo no intervalo de 0 a 32.767. Um valor 0 indica que o mapeamento sequencial não é usado. O valor de 1 assegura que elementos opcionais sejam mapeados de forma sequencial. Se o *value* do atributo **maxOccurs** for maior do que o *value* de **INLINE-MAXOCCURS-LIMIT**, o mapeamento baseado em contêiner será usado; caso contrário, o mapeamento sequencial será usado.

Ao decidir se deseja que listas de repetição variável sejam mapeadas sequencialmente, considere o comprimento de um único item de dados recorrentes. Se poucas instâncias de comprimento longo ocorrerem, o mapeamento baseado em contêiner é preferível; se muitas instâncias de comprimento curto ocorrerem, o mapeamento sequencial é preferível.

LANG = COBOL

Especifica que a linguagem de programação da estrutura de linguagem de alto nível é COBOL.

LANG = PLI-ENTERPRISE

Especifica que a linguagem de programação da estrutura de linguagem de alto nível é Enterprise PL/I.

LANG = PLI-OTHER

Especifica que a linguagem de programação da estrutura de linguagem de alto nível é um nível de PL/I diferente de Enterprise PL/I.

LANG = C

Especifica que a linguagem de programação da estrutura de linguagem de alto nível é C.

LANG = CPP

Especifica que a linguagem de programação da estrutura de linguagem de alto nível é C++.

LOGFILE = value

O nome completo do arquivo do z/OS UNIX no qual o DFHWS2LS grava seu log de atividades e informações de rastreamento. O DFHWS2LS criará o arquivo, mas não a estrutura de diretório, se ele ainda não existir.

Normalmente, você não usa esse arquivo, mas ele poderá ser solicitado pela organização de serviço da IBM caso você encontre problemas com o DFHWS2LS.

MAPPING-LEVEL = { 1.0 | 1.1 | 1.2 | 2.0 | 2.1 | 2.2 | 3.0 | 4.0 | 4.1 }

Especifica o nível de mapeamento que o DFHWS2LS usa ao gerar o arquivo de ligação de serviço da web e a estrutura de linguagem. Você pode selecionar estas opções:

- 1.0** O arquivo de ligação de serviço da web e a estrutura de linguagem são gerados usando os níveis de mapeamento do CICS TS 3.1.
- 1.1** Os atributos XML e tipos de dados <list> e <union> são mapeados para a estrutura de linguagem. Dados de caractere e binários que possuem um comprimento máximo de mais de 32.767 bytes são mapeados para um contêiner. O nome do contêiner é criado na estrutura de linguagem.
- 1.2** Use os parâmetros **CHAR-VARYING** e **CHAR-VARYING-LIMIT** para controlar como os dados de caractere são mapeados e processados no tempo de execução. Se você não especificar um desses parâmetros, dados binários e de caractere que possuem um comprimento máximo de

menos de 32.768 bytes serão mapeados para uma estrutura VARYING para todas as linguagens, exceto C ++, no qual os dados de caractere são mapeados para uma sequência com terminação nula.

- 2.0 Use este nível de mapeamento em uma região do CICS TS 3.2 ou posterior para tirar vantagem dos aprimoramentos no mapeamento entre a estrutura de linguagem e o arquivo de ligação de serviços da web.
- 2.1 Use este nível de mapeamento com uma região do CICS TS 3.2 que tenha o APAR PK59794 aplicado ou qualquer região mais recente do que o CICS TS 3.2 para suporte de <xsd:any> e xsd:anyType, a opção para mapear conteúdo de repetição variável sequencial com o parâmetro **INLINE-MAXOCCURS-LIMIT** e o suporte para minOccurs="0 " no <xsd:sequence>, <xsd:choice> e <xsd:all>.
- 2.2 Use este nível de mapeamento com uma região do CICS TS 3.2 que tenha o APAR PK69738 aplicado ou com qualquer região mais recente do que o CICS TS 3.2. Ele fornece o suporte a seguir:
 - Elementos com valores fixos
 - Suporte aprimorado para elementos <xsd:choice>
 - Tipos de dados abstratos
 - Elementos abstratos
 - Grupos de Substituição
- 3.0 Use este nível de mapeamento com uma região do CICS TS 4.1 ou mais recente. Neste nível de mapeamento é possível transformar registros de data e hora no formato CICS ABSTIME.
- 4.0 Use este nível de mapeamento com uma região do CICS TS 5.2 ou mais recente quando desejar usar UTF-16.
- 4.1 Para suporte de matriz truncável, use este nível de mapeamento com uma região CICS TS V5.2 que tenha o APAR PI67641 aplicado ou qualquer região CICS TS V5.3 ou mais recente...

Para obter mais informações sobre níveis de mapeamento, consulte Níveis de mapeamento para os assistentes CICS.

MAPPING-OVERRIDES = { SAME-AS-MAPPING-LEVEL | HYPHENS-AS-UNDERScores | LESS-DUP-NAMES | NO-ARRAY-NAME-INDEXING | UNDERScores-AS-HYPHENS }

Especifica se o comportamento padrão é substituído para o nível de mapeamento especificado ao gerar estruturas de linguagem.

Nota: Qualquer uma das subopções pode ser usada em uma lista delimitada por vírgulas. As opções não são mutuamente exclusivas; elas são combinatórias e não ordenadas.

SAME-AS-MAPPING-LEVEL

Este parâmetro gera estruturas de linguagem no mesmo estilo que o nível de mapeamento. Este é o padrão.

HYPHENS-AS-UNDERScores

Para PL/I somente. Este parâmetro converte quaisquer hifens no documento WSDL em sublinhados em vez do caractere X, para melhorar a capacidade de leitura das estruturas de linguagem PL/I geradas. Para obter mais informações, consulte Esquema XML para Mapeamento de PL/I.

LESS-DUP-NAMES

Este parâmetro gera nomes de campo de estrutura não estruturais com `_value` no final do nome para ativar a referência direta ao campo. Por exemplo, na estrutura de linguagem PLI a seguir, quando `MAPPING-OVERRIDES=LESS-DUP-NAMES` é especificado, o campo de nível 12 `streetName` é sufixado com `_value` :

```
09 streetName,  
12 streetName CHAR(255) VARYING  
UNALIGNED,  
12 filler BIT (7),  
12 attr_nil_streetName_value BIT (1),
```

A estrutura resultante é a seguinte:

```
09 streetName,  
12 streetName_value CHAR(255) VARYING  
UNALIGNED,  
12 filler BIT (7),  
12 attr_nil_streetName_value BIT (1),
```

NO-ARRAY-NAME-INDEXING

Apenas para Enterprise PL/I. Assegura que os nomes de campos dentro de uma matriz sejam exclusivos somente dentro do escopo da estrutura de nível superior.

UNDERScores-AS-HYPHENS

Esta opção é ativada automaticamente no nível de mapeamento 4.0.

Apenas para COBOL. Este parâmetro converte quaisquer sublinhados no documento WSDL em hifens, em vez do caractere `X`, para melhorar a capacidade de leitura das estruturas de linguagem COBOL geradas. Se ocorrerem quaisquer conflitos de nomes de campo, os campos serão numerados para garantir que sejam exclusivos. Para obter mais informações, consulte Esquema XML para mapeamento COBOL.

MINIMUM-RUNTIME-LEVEL = { MINIMUM | 1.0 | 1.1 | 1.2 | 2.0 | 2.1 | 2.2 | 3.0 | 4.0 | 4.1 | **CURRENT** }

Especifica o ambiente de tempo de execução mínimo do CICS no qual o arquivo de ligação de serviço da web pode ser implementado. Se você selecionar um nível que não corresponde aos outros parâmetros que você especificou, você receberá uma mensagem de erro. Você pode selecionar estas opções:

MINIMUM

O menor nível de tempo de execução possível do CICS é alocado automaticamente de acordo com os parâmetros que você especificou.

- 1.0** O arquivo de ligação de serviço da web gerado é implementado com sucesso em uma região do CICS TS 3.1 que não tem as APARs PK15904 e PK23547 aplicadas. Alguns parâmetros não estão disponíveis neste nível de tempo de execução.
- 1.1** O arquivo de ligação de serviço da web gerado é implementado com sucesso em uma região do CICS TS 3.1 que tem pelo menos o APAR PK15904 aplicado. É possível usar um nível de mapeamento 1.1 ou anterior para o parâmetro `MAPPING-LEVEL`. Alguns parâmetros não estão disponíveis neste nível de tempo de execução.
- 1.2** O arquivo de ligação de serviço da web gerado é implementado com sucesso em uma região do CICS TS 3.1 que tem ambos os APARs, PK15904 e PK23547, aplicados. É possível usar um nível de

mapeamento 1.2 ou anterior para o parâmetro **MAPPING-LEVEL**. Alguns parâmetros não estão disponíveis neste nível de tempo de execução.

- 2.0 O arquivo de ligação de serviço da web gerado é implementado com sucesso em um CICS TS 3.2 ou em uma região posterior. É possível usar um nível de mapeamento de 2.0 ou anterior para o parâmetro **MAPPING-LEVEL**. Alguns parâmetros não estão disponíveis neste nível de tempo de execução.
- 2.1 O arquivo de ligação de serviço da web gerado é implementado com sucesso em uma região do CICS TS 3.2 que tem o APAR PK59794 aplicado ou em qualquer região posterior ao CICS TS 3.2. É possível usar um nível de mapeamento 2.1 ou anterior para o parâmetro **MAPPING-LEVEL**. Alguns parâmetros não estão disponíveis neste nível de tempo de execução.
- 2.2 O arquivo de ligação de serviço da web gerado é implementado com sucesso em uma região do CICS TS 3.2 que tem o APAR PK69738 aplicado ou em qualquer região posterior ao CICS TS 3.2. Com este nível de tempo de execução, é possível usar um nível de mapeamento 2.2 ou anterior para o parâmetro **MAPPING-LEVEL**. Alguns parâmetros não estão disponíveis neste nível de tempo de execução.
- 3.0 O arquivo de ligação de serviço da web gerado é implementado com sucesso em um CICS TS 4.1 ou em uma região posterior. Com este nível de tempo de execução, é possível usar um nível de mapeamento 3.0 ou anterior para o parâmetro **MAPPING-LEVEL**. Alguns parâmetros não estão disponíveis neste nível de tempo de execução.
- 4.0 O arquivo de ligação de serviço da web gerado é implementado com sucesso em um CICS TS 5.2 ou em uma região posterior. Com este nível de tempo de execução, é possível usar um nível de mapeamento 4.0 ou anterior para o parâmetro **MAPPING-LEVEL**. É possível utilizar qualquer parâmetro opcional neste nível.
- 4.1 O arquivo de ligação de serviços da web gerado é implementado com sucesso em uma região CICS V5.2 que possui o APAR PI67641 aplicado ou em qualquer região CICS V5.3 ou mais recente. Com este nível de tempo de execução, é possível utilizar um nível de mapeamento 4.1 ou anterior para o parâmetro **MAPPING-LEVEL**.

CURRENT

O arquivo de ligação de serviço da web gerado é implementado com sucesso em uma região CICS no mesmo nível de tempo de execução que você está usando para gerar o arquivo de ligação de serviço da web.

NAME-TRUNCATION = { LEFT | RIGHT }

Especifica se os nomes de elementos XML são truncados a partir da esquerda ou da direita. O assistente de serviço da web do CICS trunca nomes de elemento XML para o comprimento apropriado para a linguagem de alto nível especificada; por padrão, os nomes são truncados a partir da direita.

OPERATIONS = value

Para aplicativos do solicitante de serviço da web, especifica um subconjunto de elementos `<wsdl:Operation>` válidos da descrição de serviços da web que é usado para gerar o arquivo de ligação de serviço da web. Cada elemento

<wsdl:Operation> é separado por um espaço; a lista pode se estender por mais de uma linha se necessário. É possível usar esse parâmetro para documentos WSDL 1.1 e WSDL 2.0.

PDSCP = *value*

Especifica a página de códigos usada nos membros do conjunto de dados particionados especificados nos parâmetros **REQMEM** e **RESPMEM**, em que *value* é um número de CCSID ou um número da página de código Java. Se este parâmetro não for especificado, a página de códigos do z/OS UNIX System Services será usada. Por exemplo, você pode especificar **PDSCP** = 037.

PDSLIB = *value*

Especifica o nome do conjunto de dados particionados que contém a linguagem de alto nível gerada. Os membros do conjunto de dados usados para a solicitação e resposta são especificados nos parâmetros **REQMEM** e **RESPMEM**, respectivamente.

PDSMEM = *value*

Especifica um prefixo de 1 a 6 caracteres que o DFHWS2LS usa para gerar os nomes dos membros do conjunto de dados particionados que conterão as estruturas de linguagem de alto nível para tipos de dados abstratos. Ele gera o nome do membro anexando um número de 2 dígitos no prefixo.

Use este parâmetro no nível de mapeamento 2.2 ou superior para nomear as estruturas de linguagem associadas com os tipos de dados abstratos. Se o parâmetro **PDSMEM** for omitido, as estruturas de linguagem para tipos de dados abstratos serão nomeadas usando o valor no parâmetro **REQMEM**.

PGMINT = { **CHANNEL** | **COMMAREA** }

Para um provedor de serviços, especifica como o CICS transmite dados ao programa de aplicativo de destino:

CHANNEL

CICS usa uma interface do canal para transmitir dados ao programa de aplicativo de destino.

COMMAREA

CICS usa uma área de comunicação para transmitir dados ao programa de aplicativo de destino.

Este parâmetro é ignorado quando a saída de DFHWS2LS é usada em um solicitante de serviço.

Quando o programa de aplicativo de destino tiver processado a solicitação, ele deverá usar o mesmo mecanismo para retornar a resposta. Se a solicitação foi recebida em uma área de comunicação, a resposta deve ser retornada na área de comunicação; se a solicitação foi recebida em um contêiner, a resposta deve ser retornada em um contêiner. O comprimento da área de comunicação ou contêiner que o CICS transmite para o programa de aplicativo de destino é o maior dos comprimentos da área ou do contêiner de comunicação da solicitação e da área ou contêiner de comunicação de resposta.

PGMNAME = *value*

Especifica o nome de um recurso do PROGRAMA CICS.

Quando DFHWS2LS é usado para gerar um arquivo de ligação de serviço da web que será usado em um provedor de serviços, deve-se fornecer esse parâmetro. Ele especifica o nome do recurso do programa de aplicativo que é exposto como um serviço da web.

Quando DFHWS2LS for usado para gerar um arquivo de ligação de serviço da web que será usado em um solicitante de serviço, omita este parâmetro.

REQMEM = *value*

Especifica um prefixo de 1 a 6 caracteres que o DFHWS2LS usa para gerar os nomes dos membros do conjunto de dados particionados que conterão as estruturas de linguagem de alto nível para a solicitação de serviço da web:

- Para um provedor de serviços, a solicitação de serviço da web é a entrada para o programa de aplicativo.
- Para um solicitante de serviço, a solicitação de serviço da web é a saída do programa de aplicativo.

DFHWS2LS gera um membro do conjunto de dados particionados para cada operação. Ele gera o nome do membro anexando um número de 2 dígitos no prefixo.

Embora este parâmetro seja opcional, você deve especificá-lo se a descrição de serviços da web contém uma definição de uma solicitação.

RESPMEM = *value*

Especifica um prefixo de 1 a 6 caracteres que o DFHWS2LS usa para gerar os nomes dos membros do conjunto de dados particionados que conterão as estruturas de linguagem de alto nível para a resposta de serviço da web:

- Para um provedor de serviços, a resposta de serviço da web é a saída do programa de aplicativo.
- Para um solicitante de serviço, a resposta do serviço da web é a entrada para o programa de aplicativo.

DFHWS2LS gera um membro do conjunto de dados particionados para cada operação. Ele gera o nome do membro anexando um número de 2 dígitos no prefixo.

Omita esse parâmetro se nenhuma resposta estiver envolvida; ou seja, para mensagens unidirecionais.

SSL-KEYSTORE = *value*

Este parâmetro opcional especifica o local completo do arquivo de armazenamento de chaves.

Use este parâmetro se desejar que o assistente de serviços da web use a criptografia secure sockets layer (SSL) para se comunicar em uma rede com um IBM WebSphere Service Registry and Repository (WSRR).

SSL-KEYPWD = *value*

Este parâmetro opcional especifica a senha para o armazenamento de chaves.

Use este parâmetro se desejar que o assistente de serviços da web use a criptografia secure sockets layer (SSL) para se comunicar em uma rede com um IBM WebSphere Service Registry and Repository (WSRR).

SSL-TRUSTSTORE = *value*

Este parâmetro opcional especifica o local completo do arquivo de armazenamento confiável.

Use este parâmetro se desejar que o assistente de serviços da web use a criptografia secure sockets layer (SSL) para se comunicar em uma rede com um IBM WebSphere Service Registry and Repository (WSRR).

SSL-TRUSTPWD = *value*

Este parâmetro opcional especifica a senha para o armazenamento confiável.

Use este parâmetro se desejar que o assistente de serviços da web use a criptografia secure sockets layer (SSL) para se comunicar em uma rede com um IBM WebSphere Service Registry and Repository (WSRR).

STRUCTURE = (*request* , *response*)

Apenas para C e C++, especifica como os nomes das estruturas de solicitação e resposta são gerados.

As estruturas de solicitação e resposta geradas recebem nomes de *request nn* e *response nn* em que *nn* é um sufixo numérico que é gerado para distinguir as estruturas para cada operação.

Se um ou ambos os nomes forem omitidos, as estruturas terão o mesmo nome que os nomes de membros do conjunto de dados particionados gerados a partir dos parâmetros **REQMEM** e **RESPMEM** que você especifica.

SYNCONRETURN = { **NO** | **YES** }

Especifica se o serviço da web remoto pode emitir um ponto de sincronização.

NO O serviço da web remoto não pode emitir um ponto de sincronização. Este valor é o padrão. Se o serviço da web remoto emitir um ponto de sincronização, ele falhará com um encerramento anormal de ADPL.

YES O serviço da web remoto pode emitir um ponto de sincronização. Se você selecionar YES, a tarefa remota será confirmada como uma unidade de trabalho separada quando o controle retornar do serviço da web remoto. Se o serviço da web remoto atualizar um recurso recuperável e ocorrer uma falha após ele retornar, a atualização para esse recurso não poderá ser restaurada.

TRANSACTION = *name*

Em um provedor de serviços, este parâmetro especifica o nome com 1 a 4 caracteres de uma transação de alias que pode iniciar o pipeline. O valor desse parâmetro é usado para definir o atributo TRANSACTION do recurso URIMAP quando ele é criado automaticamente usando o comando de varredura **PIPELINE**.

Caracteres aceitáveis:

A-Z a-z 0-9 \$ @ # _ < >

URI = *value*

Em um provedor de serviços, este parâmetro especifica o URI relativo que um cliente usa para acessar o serviço da web. O CICS usa o valor especificado quando ele gera um recurso URIMAP a partir do arquivo de ligação de serviço da web criado por DFHWS2LS. O parâmetro especifica o componente de caminho do URI ao qual a definição de URIMAP se aplica.

Em um solicitante de serviço, o URI do serviço da web de destino *não* é especificado com este parâmetro. O CICS não gera um recurso URIMAP para um solicitante de serviço. É possível definir seu próprio recurso URIMAP para solicitantes de serviço usarem ao fazerem solicitações do cliente ao URI do serviço da web de destino. Quando um solicitante de serviço emite o comando, **INVOKE SERVICE** o CICS usa o local soap:address a partir do wsdl:port especificado na descrição de serviços da web, se presente. É possível substituir isso e especificar um URI diferente usando as opções URIMAP ou URI no comando **INVOKE SERVICE**.

USERID = *id*

Em um provedor de serviços, este parâmetro especifica um ID do usuário com 1 a 8 caracteres, que pode ser usado por qualquer Web client. Para uma resposta gerada por aplicativo ou um serviço da web, a transação de alias é conectada com este ID do usuário. O valor desse parâmetro é usado para definir o atributo USERID do recurso URIMAP quando ele é criado automaticamente usando o comando de varredura **PIPELINE**.

Caracteres aceitáveis:

A-Z a-z 0-9 \$ @ #

WIDE-COMP3 = { FULL | NO | YES }

Controla o tamanho máximo do comprimento variável decimal compactado na estrutura de linguagem COBOL ou PL/I gerada.

FULL Para COBOL e PL/I. DFHJS2LS gera um campo decimal compactado que é grande o suficiente para conter todos os valores válidos. O tamanho máximo são 31 dígitos. Este é o padrão.

NO Apenas para COBOL. DFHJS2LS limita o comprimento variável decimal compactado a 18 ao gerar a estrutura de linguagem COBOL tipo COMP-3. Se o tamanho decimal compactado for maior que 18, a mensagem DFHPI9022W será emitida para indicar que o tipo especificado está sendo limitado a um total de 18 dígitos.

YES Apenas para COBOL. DFHJS2LS suporta o tamanho máximo de 31 ao gerar a estrutura de linguagem COBOL tipo COMP-3.

Nota: As opções NO e YES geram campos que são incapazes de representar todos os valores válidos; a opção FULL evita esse problema. No entanto, a opção FULL permite que alguns valores inválidos sejam representados no campo decimal compactado. Por exemplo, se um esquema indica que há um máximo de cinco dígitos e um máximo de dois dígitos fracionários, a opção FULL gerará um campo decimal compactado que permite sete dígitos e isso permite espaço para valores válidos como 25000 e 999,99, mas também fornece espaço para alguns valores inválidos como 9999,99. Quando você usar a opção FULL, tome cuidado para não gerar valores inválidos em dados do aplicativo.

WSADDR-EPR-ANY = { TRUE | FALSE }

Especifica se o CICS transforma uma referência do terminal do WS-Addressing (EPR) em suas partes componentes nas estruturas de linguagem ou trata o EPR como um tipo <xsd:any>. Tratar o EPR como um tipo <xsd:any> significa que a API **WSACONTEXT BUILD** pode usar o XML do EPR diretamente.

FALSE

DFHWS2LS se comporta tipicamente, transformando o XML em uma estrutura de linguagem de alto nível.

TRUE Configurar essa opção como TRUE significa que no tempo de execução o CICS trata o EPR inteiro como um tipo <xsd:any> e coloca o XML do EPR em um contêiner que pode ser referenciado pelo aplicativo. O aplicativo pode usar o XML do EPR com a API **WSACONTEXT BUILD** para construir um EPR no contexto de endereçamento.

Este parâmetro está disponível somente no nível de tempo de execução 3.0 em diante.

WSBIND = *value*

O nome completo do z/OS UNIX do arquivo de ligação de serviço da web. O DFHWS2LS criará o arquivo, mas não a estrutura de diretório, se ele ainda não existir. A extensão do arquivo é padronizada como *.wsbind*.

WSDL = *value*

O nome completo do arquivo do z/OS UNIX que contém a descrição de serviços da web. O nome do arquivo é restrito a caracteres que são válidos para uma URL e, em particular, não pode conter nenhum caractere #. Se você

estiver usando WSRR para recuperar o documento WSDL, este parâmetro especifica o local no sistema de arquivos no qual uma cópia local do documento WSDL será gravada.

WSDL-SERVICE = *value*

Especifica o elemento `wsdl:Service` que é usado quando a descrição de serviços da web contém mais de um elemento `Service` para um elemento `Binding`. Se você especificar um valor para o parâmetro **BINDING**, o elemento `Service` que você especificar para este parâmetro deverá ser consistente com o elemento `Binding` especificado. É possível usar esse parâmetro com documentos WSDL 1.1 ou WSDL 2.0.

WSRR-NAME = *value*

Especifica o nome do documento WSDL para recuperar a partir do WSRR. Use esse parâmetro somente quando o parâmetro **WSRR-SERVER** for especificado.

WSRR-NAMESPACE = *value*

Especifica o namespace do documento WSDL a ser recuperado a partir do WSRR. Opcionalmente, é possível usar este parâmetro quando o parâmetro **WSRR-SERVER** é especificado para qualificar completamente o nome do documento WSDL especificado no parâmetro **WSRR-NAME**.

WSRR-PASSWORD = *value*

Use este parâmetro opcional se você deve inserir uma senha para acessar o WSRR.

Se o parâmetro **WSRR-USERNAME** for especificado, você também deverá especificar esse parâmetro.

Use esse parâmetro somente quando o parâmetro **WSRR-SERVER** for especificado.

WSRR-SERVER = { *domain name* : *port number* | *IP address* : *port number* }

Use este parâmetro para especificar o local do servidor IBM WebSphere Service Registry and Repository (WSRR). Se este parâmetro for especificado, a validação de parâmetro WSRR será usada.

WSRR-USERNAME = *value*

Use este parâmetro opcional se for necessário especificar um nome de usuário para acessar o WSRR. Esse nome de usuário é usado pelo WSRR para configurar a propriedade de proprietário.

Use esse parâmetro somente quando o parâmetro **WSRR-SERVER** for especificado.

WSRR-VERSION = *value*

Especifica a versão do documento WSDL para recuperar a partir do WSRR. É possível usar esse parâmetro somente quando o parâmetro **WSRR-SERVER** é especificado.

XML-ONLY = { **TRUE** | **FALSE** }

Especifica se o CICS transforma o XML na mensagem SOAP em dados do aplicativo ou não. Use o parâmetro **XML-ONLY** para gravar aplicativos de serviço da web que processam o XML eles mesmos.

TRUE O CICS não executa nenhuma transformação para o XML. O solicitante de serviço ou o aplicativo do provedor deve trabalhar com o conteúdo do contêiner DFHWS-BODY diretamente para mapear dados entre XML e a linguagem de alto nível.

FALSE

O CICS transforma o XML em uma linguagem de alto nível.

Este parâmetro está disponível somente no nível de tempo de execução 2.1 em diante.

Outras informações

- O ID do usuário sob o qual o DFHLS2SC é executado deve ser configurado para usar o UNIX System Services. O ID do usuário deve ter permissão de leitura para a estrutura de arquivo e bibliotecas PDS do CICS z/OS UNIX e permissão de gravação para os diretórios especificados nos parâmetros **LOGFILE**, **WSBIND** e **WSDL**.
- O ID do usuário deve ter uma alocação de armazenamento suficientemente grande para executar Java.
- A JCL possui um comprimento de parâmetro máximo de 100 caracteres. Isso pode ser aumentado usando a instrução **STDPARM**. Para obter mais informações, consulte Guia do usuário do z/OS UNIX System Services.

Exemplo

```
//WS2LS JOB '  
accounting information  
'  
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  
/*
```

Parâmetros Requeridos por DFHWS2LS para Suportar WS-Addressing:

Quando você configura seu WSDL for Web Services Addressing, deve-se configurar os parâmetros **MINIMUM-RUNTIME** e **MAPPING-LEVEL** no assistente de serviços da web, DFHWS2LS, como um valor de 3.0 ou mais alto. Você também pode desejar considerar configurar o parâmetro **WSADDR-EPR-ANY** como TRUE.

Configure o parâmetro **MINIMUM-RUNTIME** no assistente de serviços da web, DFHWS2LS, como 3.0 ou superior. Um nível de tempo de execução de pelo menos 3.0 assegura que quaisquer arquivos WSBinding que o assistente gera suportem totalmente o endereçamento de serviço da web e possam interoperar com outras plataformas de serviços da web.

Configure o parâmetro **MAPPING-LEVEL** no assistente de serviços da web, DFHWS2LS, como 3.0 ou superior.

Se tiver qualquer elemento do tipo `wsa:EndpointReferenceType` nas mensagens de solicitação ou resposta definidas em seu documento WSDL, e desejar usar estes

elementos como entrada no comando da API **WSACONTEXT BUILD** no tempo de execução, configure o parâmetro **WSADDR-EPR-ANY** como TRUE. A configuração do parâmetro **WSADDR-EPR-ANY** como TRUE indica que o CICS não deve transformar o EPR em uma estrutura de linguagem no tempo de execução; em vez disso, o CICS deve tratar os dados de EPR como um elemento `<xsd:any>` e armazená-los em um contêiner nomeado.

Este fragmento de WSDL de exemplo mostra um MAP `<wsa:To>` sendo transmitido como um elemento do 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>
```

Quando o elemento, `<xs:element name="myEpr" type="wsa:EndpointReferenceType"/>` **1**, for processado por DFHWS2LS com o parâmetro **WSADDR-EPR-ANY** configurado como TRUE, os dados do elemento `myEpr` serão armazenados em um contêiner nomeado como um elemento `<xsd:any>` no tempo de execução e um ponteiro no contêiner incluído na estrutura de linguagem gerada.

Por exemplo, a estrutura de linguagem COBOL gerada por DFHWS2LS para o elemento `myEpr` é mostrada aqui:

```
09 myEpr.
   12 myEpr-xml-cont          PIC X(16).
   12 myEpr-xmlns-cont       PIC X(16).
```

O contêiner `myEpr-xml-cont` armazena o nome do contêiner que contém os dados `myEpr`. `myEpr-xmlns-cont` é um contêiner opcional que é preenchido com quaisquer declarações de namespace XML que estão no escopo.

Níveis de mapeamento para os assistentes do CICS

Um mapeamento é o conjunto de regras que especifica como as informações são convertidas entre as estruturas de linguagem e esquemas XML. Para se beneficiar dos mapeamentos mais sofisticados disponíveis, deve-se configurar o parâmetro **MAPPING-LEVEL** nos assistentes do CICS com o nível mais recente.

Cada nível de mapeamento herda a função do mapeamento anterior, em que o nível mais alto de mapeamento oferece os melhores recursos disponíveis. O nível de mapeamento mais alto fornece mais controle sobre a conversão de dados no tempo de execução e remove restrições no suporte para determinados tipos de dados e elementos XML.

É possível configurar o parâmetro **MAPPING-LEVEL** para um nível anterior se você deseja reimplementar aplicativos que foram ativados anteriormente nesse nível.

Nível de mapeamento 4.1

O nível de mapeamento 4.1 é compatível com uma região CICS TS 5.3 ou com a região CICS TS 5.2 com a APAR PI67641 aplicada e superiores.

O nível de mapeamento 4.1 é incluído nos Assistentes de Serviços da Web, nos Assistentes XML e Assistentes JSON. Este nível de mapeamento implementa mapeamentos melhorados para matrizes simples gerados de modo ascendente a partir de copybooks existentes; ele também inclui a capacidade do CICS de detectar automaticamente armazenamento final não inicializado em matrizes e de omitir esses registros no formulário XML/JSON gerado. Para obter mais informações, consulte “Mapeamento de esquema COBOL para JSON” na página 434.

DFHLS2WS, DFHWS2LS, DFHLS2SC, DFHSC2LS, DFHJS2LS e DFHLS2JS suportam os parâmetros **TRUNCATE-NULL-ARRAYS** e **TRUNCATE-NULL-ARRAY-VALUES**.

Se você especificar qualquer valor para **TRUNCATE-NULL-ARRAY-VALUESS**, também deverá especificar **TRUNCATE-NULL-ARRAYS=ENABLED**.

Nível de mapeamento 4.0

O nível de mapeamento 4.0 é compatível com uma região do CICS TS 5.2.

No nível de mapeamento 4.0 e superior, DFHLS2SC e DFHLS2WS suportam a cláusula COBOL OCCURS DEPENDING ON e suportam o mapeamento de matrizes de caracteres COBOL para sequências XML. É possível configurar este comportamento usando o parâmetro **CHAR-OCCURS** nos assistentes do CICS.

- Deve-se especificar o parâmetro **DATA-TRUNCATION=ENABLED**.
- OCCURS DEPENDING ON complexo não é suportado. Esta limitação significa que OCCURS DEPENDING ON é suportado somente para o último campo de uma estrutura.
- O CICS não suporta nomes qualificados (usando a palavra-chave 'OF') como o destino de uma cláusula OCCURS DEPENDING ON, por exemplo FIELD1 OF STRUCTURE1.
- O CICS não suporta a palavra-chave UNBOUNDED. Deve-se especificar o tamanho máximo da tabela que é esperado pelo aplicativo.

No nível de mapeamento 4.0 e superior, os serviços da web do CICS suportam a conversão de dados do aplicativo que são codificados usando UTF-16 Unicode.

- Quando você usa LS2WS ou LS2SC, é possível ativar este comportamento usando tipos de dados específicos de linguagem para UTF-16.
- Quando você usa WS2LS ou SC2LS, é possível ativar este comportamento ao configurar CCSID=1200.

- CICS suporta somente uma única página de códigos Unicode, “ UTF-16BE com IBM Private Use Area ” (CCSID 1200).
- A conversão de dados do aplicativo que são codificados usando UTF-8 não é suportada.

Nota: DFHLS2WS e DFHLS2SC não suportam a cláusula COBOL GROUP USAGE NATIONAL.

Nível de mapeamento 3.0 e superior

O nível de mapeamento 3.0 é compatível com uma região do CICS TS 4.1 e superior.

Este nível de mapeamento fornece o suporte a seguir:

- DFHSC2LS e DFHWS2LS mapeiam tipos de dados xsd:dateTime para o formato CICS ASKTIME.
- O DFHLS2WS pode gerar um documento WSDL e a ligação de serviço da web a partir de um aplicativo que usa muitos contêineres em vez de somente um contêiner.
- Tolerando dados truncados que são descritos por uma estrutura de dados de comprimento fixo. É possível configurar este comportamento usando o parâmetro **DATA-TRUNCATION** nos assistentes do CICS.

Nível de Mapeamento 2.2 e Superior

O nível de mapeamento 2.2 é compatível com uma região do CICS TS 3.2, com o APAR PK69738 aplicado, e superior.

No nível de mapeamento 2.2 e superior, DFHSC2LS e DFHWS2LS suportam os mapeamentos XML a seguir:

- Valores fixos para elementos
- Grupos de Substituição
- Tipos de dados abstratos
- Elementos <sequence> de esquema XML podem aninhar dentro de elementos <choice>

DFHSC2LS e DFHWS2LS fornecem suporte aprimorado para os mapeamentos XML a seguir:

- Elementos abstratos
- Elementos <choice> do esquema XML

Nível de Mapeamento 2.1 e Superior

O nível de mapeamento 2.1 é compatível com uma região do CICS TS 3.2, com o APAR PK59794 aplicado, e superior.

Este nível de mapeamento inclui maior controle sobre a maneira como o conteúdo variável é manipulado com o novo parâmetro **INLINE-MAXOCCURS-LIMIT** e novos valores no parâmetro **CHAR-VARYING**.

No nível de mapeamento 2.1 e superior, DFHSC2LS e DFHWS2LS oferecem o novo suporte aprimorado para mapeamentos XML a seguir:

- O elemento <any> do esquema XML

- O tipo `xsd:anyType`
- Tolerância de elementos abstratos
- O parâmetro **INLINE-MAXOCCURS-LIMIT**
- O atributo `minOccurs`

O parâmetro **INLINE-MAXOCCURS-LIMIT** especifica se listas de repetição variável são mapeadas sequencialmente. Para obter mais informações sobre o mapeamento de conteúdo de repetição variável sequencial, consulte Matrizes Variáveis de Elementos.

O suporte para o atributo `minOccurs` é aprimorado para os elementos `<sequence>`, `<choice>` e `<all>` do esquema XML. Se `minOccurs="0"`, o assistente do CICS trata esses elementos como se o atributo `minOccurs="0"` também fosse um atributo de todos os seus elementos filhos.

No nível de mapeamento 2.1 e superior, DFHLS2SC e DFHLS2WS suportam os mapeamentos XML a seguir:

- Campos FILLER em COBOL e PL/I são ignorados
- Um valor de COLLAPSE para o parâmetro **CHAR-VARYING**
- Um valor de BINARY para o parâmetro **CHAR-VARYING**

Campos FILLER em COBOL e PL/I são ignorados; eles não aparecem no esquema XML gerado e um intervalo apropriado é deixado nas estruturas de dados no tempo de execução.

COLLAPSE faz com que o CICS ignore espaços à direita em campos de texto.

BINARY fornece suporte para campos binários. Este valor é útil quando você está convertendo COBOL em um esquema XML. Essa opção está disponível somente em matrizes de caracteres SBCS e permite que a matriz seja mapeada para campos `xsd:base64Binary` de comprimento fixo em vez de campos `xsd:string`.

Nível de Mapeamento 1.2 e Superior

O nível de mapeamento 1.2 é compatível com uma região CICS TS 3.1 e superior.

Há maior controle disponível na maneira como os caracteres e dados binários são transformados no tempo de execução com esses parâmetros adicionais nas ferramentas em lote:

- **CHAR-VARYING**
- **CHAR-VARYING-LIMIT**
- **CHAR-MULTIPLIER**
- **DEFAULT-CHAR-MAXLENGTH**

Se você decidir usar o parâmetro **CHAR-MULTIPLIER** em DFHSC2LS ou DFHWS2LS, as regras a seguir se aplicarão após o valor desse parâmetro ser usado para calcular a quantidade de espaço que é necessário para dados de caractere.

- DFHSC2LS e DFHWS2LS fornecem estes mapeamentos:
 - Tipos de dados de caractere de comprimento variável que possuem um comprimento máximo de mais de 32767 bytes são mapeados para um contêiner. É possível usar o parâmetro **CHAR-VARYING-LIMIT** para configurar um limite inferior. Um campo de 16 bytes é criado na estrutura de linguagem

para armazenar o nome do contêiner. No tempo de execução, os dados de caracteres são armazenados em um contêiner e o nome do contêiner é colocado na estrutura de linguagem.

- Os tipos de dados de caractere de comprimento variável que possuem um comprimento máximo de menos de 32.768 bytes são mapeados para uma estrutura VARYING para todas as linguagens, exceto C/C++ e Enterprise PL/I. No C/C++, esses tipos de dados são mapeados para sequências com terminação nula e no Enterprise PL/I esses tipos de dados são mapeados para estruturas VARYINGZ. É possível usar o parâmetro **CHAR-VARYING** para selecionar o modo como os dados de caractere de comprimento variável são mapeados.
- Dados binários de comprimento variável que possuem um comprimento máximo de menos de 32768 bytes mapeiam para uma estrutura VARYING para todas as linguagens. Se o comprimento máximo for igual ou maior do que 32.768 bytes, os dados serão mapeados para um contêiner. Um campo de 16 bytes é criado na estrutura de linguagem para armazenar o nome do contêiner. No tempo de execução, os dados binários são armazenados em um contêiner e o nome do contêiner é colocado na estrutura de linguagem.

Se você tiver tipos de dados de caractere no esquema XML que não possuem um comprimento associado a eles, poderá designar um comprimento padrão usando o parâmetro **DEFAULT-CHAR-MAXLENGTH** em DFHWS2LS ou DFHSC2LS.

DFHLS2SC e DFHLS2WS fornecem estes mapeamentos:

- Os campos de caracteres mapeiam para um tipo de dados `xsd:string` e podem ser processados como campos de comprimento fixo ou sequências com terminação nula no tempo de execução. É possível usar o parâmetro **CHAR-VARYING** para selecionar o modo como os dados de caractere de comprimento variável são manipulados no tempo de execução para todas as linguagens, exceto PL/I.
- Os tipos de dados `Base64Binary` mapeiam para um contêiner se o comprimento máximo dos dados é maior do que 32767 bytes ou quando o comprimento não está definido. Se o comprimento dos dados é 32767 ou menos, o tipo de dados `base64Binary` é mapeado para uma estrutura VARYING para todas as linguagens.

Nível de Mapeamento 1.1 e Superior

O nível de mapeamento 1.1 é compatível com uma região do CICS TS 3.1 e superior.

Este nível de mapeamento fornece mapeamento aprimorado de caractere XML e tipos de dados binários, em específico, ao mapear dados de comprimento variável que possui atributos `maxLength` e `minLength` que são definidos com valores diferentes no esquema XML. Os dados são manipulados das maneiras a seguir:

- Os tipos de dados de caractere e binários que possuem um comprimento fixo que é maior do que 16 MB mapeiam para um contêiner para todas as linguagens, exceto PL/I. No PL/I, tipos de dados de caractere e binários de comprimento fixo que são maiores do que 32767 bytes são mapeados para um contêiner. Um campo de 16 bytes é criado na estrutura de linguagem para armazenar o nome do contêiner. No tempo de execução, os dados de comprimento fixo são armazenados em um contêiner e o nome do contêiner é colocado na estrutura de linguagem.

Como os contêineres são variáveis no comprimento, os dados de comprimento fixo que são mapeados para um contêiner não são preenchidos com espaços ou nulos, ou truncados, para corresponder ao comprimento fixo especificado no esquema XML ou na descrição de serviços da web. Se o comprimento dos dados for significativo, será possível gravar seu aplicativo para verificá-lo ou ativar a validação na região CICS. Tanto a validação de SOAP como a de XML possuem um impacto significativo no desempenho.

- Os tipos de dados <list> e <union> de esquema XML são mapeados para campos de caracteres.
- Os atributos XML definidos pelo esquema são mapeados em vez de ignorados. Um máximo de 255 atributos é permitido para cada elemento XML. Para obter mais informações, consulte Suporte para atributos XML.
- O atributo `xsi:nil` é suportado. Para obter mais informações, consulte Suporte para atributos XML.

Apenas Nível de Mapeamento 1.1

O nível de mapeamento 1.1 é compatível com uma região do CICS TS 3.1 e superior.

Este nível de mapeamento fornece mapeamento aprimorado de caractere XML e tipos de dados binários, em específico, ao mapear dados de comprimento variável que possui atributos `maxLength` e `minLength` que são definidos com valores diferentes no esquema XML. Os dados são manipulados das maneiras a seguir:

- Tipos de dados binários de comprimento variável mapeiam para um contêiner. Um campo de 16 bytes é criado na estrutura de linguagem para armazenar o nome do contêiner. No tempo de execução, os dados binários são armazenados em um contêiner e o nome do contêiner é colocado na estrutura de linguagem.
- Os tipos de dados de caractere de comprimento variável que possuem um comprimento máximo maior que 32767 bytes mapeiam para um contêiner. Um campo de 16 bytes é criado na estrutura de linguagem para armazenar o nome do contêiner. No tempo de execução, os dados de caractere são armazenados em um contêiner e o nome do contêiner é colocado na estrutura de linguagem.
- Os tipos de dados de caractere e binários que possuem um comprimento fixo de menos de 16 MB mapeiam para campos de comprimento fixo para todas as linguagens, exceto PL/I. Em PL/I, os tipos de dados de caractere e binários de comprimento fixo que têm 32767 bytes ou menos mapeiam para campos de comprimento fixo.
- O CICS codifica e decodifica dados no formato `hexBinary` mas não no formato `base64Binary`. Os tipos de dados `Base64Binary` no esquema XML são mapeados para um campo na estrutura de linguagem. O tamanho do campo é calculado usando a fórmula: $4 \times (\text{ceil}(z/3))$ em que:
 - z é o comprimento do tipo de dados no esquema XML.
 - $\text{ceil}(x)$ é o menor número inteiro maior ou igual a x .

Se o comprimento de z for maior do que 24566 bytes, a estrutura de linguagem resultante falhará ao compilar. Se você tiver dados `base64Binary` que são maiores do que 24566 bytes, deverá usar um nível de mapeamento 1.2. Com o nível de mapeamento 1.2, é possível mapear os dados `base64Binary` para um contêiner em vez de usar um campo na estrutura de linguagem.

Apenas Nível de Mapeamento 1.0

O nível de mapeamento 1.0 é compatível com uma região do CICS TS 3.1 e superior.

Observe as limitações a seguir, que são modificadas nos níveis de mapeamento posteriormente:

- DFHSC2LS e DFHWS2LS mapeiam tipos de dados binários e de caractere no esquema XML para campos de comprimento fixo na estrutura de linguagem. Consulte 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>
```

Esse esquema XML parcial aparece em uma estrutura de linguagem COBOL como este exemplo:

```
15 example PIC X(33000)
```

- O CICS codifica e decodifica dados no formato hexBinary mas não no formato base64Binary. DFHSC2LS e DFHWS2LS mapeiam dados Base64Binary para um campo de caracteres de comprimento fixo cujo conteúdo deve ser codificado ou decodificado pelo programa de aplicativo.
- DFHSC2LS e DFHWS2LS ignoram atributos XML durante o processamento.
- DFHLS2SC e DFHLS2WS interpretam campos binários e de caractere na estrutura de linguagem como campos de comprimento fixo e mapeiam esses campos para elementos XML que possuem um atributo maxLength. No tempo de execução, os campos na estrutura de linguagem são preenchidos por espaços ou nulos se dados insuficientes estão disponíveis.

Linguagem de Alto Nível e Mapeamento de Esquema XML:

Utilize os assistentes do CICS para gerar mapeamentos entre as estruturas de linguagem de alto nível e os esquemas XML ou documentos WSDL. Os assistentes do CICS também geram esquemas XML ou documentos WSDL a partir de estruturas de dados de linguagem de alto nível ou vice-versa.

Os programas utilitários DFHSC2LS e DFHLS2SC são conhecidos coletivamente como o assistente XML do CICS. Os programas utilitários DFHWS2LS e DFHLS2WS são conhecidos coletivamente como o assistente de serviços da web do CICS.

- DFHLS2SC e DFHLS2WS mapeiam estruturas de linguagem de alto nível para esquemas XML e documentos WSDL respectivamente.
- DFHSC2LS e DFHWS2LS mapeiam esquemas XML e documentos WSDL para estruturas de linguagem de alto nível.

Os dois mapeamentos não são simétricos:

- Se você processar uma estrutura de dados de linguagem com DFHLS2SC ou DFHLS2WS e, em seguida, processar o esquema XML ou documento WSDL resultante com o programa utilitário complementar (respectivamente DFHSC2LS ou DFHWS2LS), não espere que a estrutura de dados final seja igual à original. No entanto, a estrutura de dados final é logicamente equivalente à original.

- Se você processar um esquema XML ou documento WSDL com DFHSC2LS ou DFHWS2LS e, em seguida, processar a estrutura de linguagem resultante com o programa utilitário complementar (respectivamente DFHLS2SC ou DFHLS2WS), não espere que o esquema XML no esquema XML ou documento WSDL final seja igual ao original.
- Em alguns casos, DFHSC2LS e DFHWS2LS geram estruturas de linguagem que não são suportadas pelo DFHLS2SC e DFHLS2WS.

Deve-se codificar estruturas de linguagem processadas por DFHLS2SC e DFHLS2WS de acordo com as regras da linguagem, conforme implementado nos compiladores de linguagem que o CICS suporta.

Mapeamento de esquema COBOL para JSON:

O programa utilitário DFHLS2JS suporta mapeamentos entre as estruturas de dados COBOL e as definições de esquema JSON.

Os nomes COBOL são convertidos em nomes JSON de acordo com as regras a seguir:

- Nomes duplicados se tornam exclusivos pela adição de um ou mais dígitos numéricos.
Por exemplo, duas instâncias de year se tornam year e year1.
- Hifens são substituídos por sublinhados. Sequências de hifens contíguos são substituídas por sublinhados contíguos.
Por exemplo, current-user--id se torna current_user_id.
- Segmentos de nomes que são delimitados por hifens e que contêm somente caracteres em maiúsculas são convertidos em minúsculas.
Por exemplo, CA-REQUEST-ID se torna ca_request_id.
- Um sublinhado inicial é incluído em nomes que iniciam com um caractere numérico.
Por exemplo, 9A-REQUEST-ID se torna _9a_request_id.

O CICS mapeia elementos de descrição de dados COBOL para elementos de esquema de acordo com a tabela a seguir. Elementos de descrição de dados COBOL que não são mostrados na tabela não são suportados pelo DFHLS2JS. As seguintes restrições também se aplicam:

- Os itens de descrição de dados com os números de nível 66 e 77 não são suportados. Os itens de descrição de dados com um número de nível 88 são ignorados.
- As seguintes cláusulas nas entradas de descrição de dados não são suportadas:
REDEFINES
RENAMES; que é o nível 66
DATE FORMAT
- As seguintes cláusulas nos itens de descrição de dados são ignoradas:
BLANK WHEN ZERO
JUSTIFIED
VALUE
- A cláusula SIGN, SIGN TRAILING, é suportada. A cláusula SIGN, SIGN LEADING, é suportada somente quando o nível de mapeamento especificado em DFHLS2JS é 1.2 ou superior.

- SEPARATE CHARACTER é suportado em um nível de mapeamento de 1.2 ou superior para ambas as cláusulas SIGN TRAILING e SIGN LEADING.
- As seguintes frases na cláusula USO não são suportadas:
 - OBJECT REFERENCE
 - POINTER
 - FUNCTION-POINTER
 - PROCEDURE-POINTER
- As frases a seguir na cláusula USAGE são suportadas em um nível de mapeamento de 1.2 ou superior:
 - COMPUTATIONAL-1
 - COMPUTATIONAL-2
- Os únicos caracteres de PICTURE suportados para os itens de descrição de dados DISPLAY e COMPUTATIONAL-5 são 9, S e Z.
- Os caracteres de PICTURE que são suportados para os itens de descrição de dados PACKED-DECIMAL são 9, S, V e Z.
- Os únicos caracteres de PICTURE que são suportados para os itens de descrição de dados numéricos editados são 9 e Z.
- Se o parâmetro MAPPING-LEVEL estiver configurado como 1.2 ou superior e o parâmetro CHAR-VARYING estiver configurado como NULL, as matrizes de caracteres serão mapeadas para uma sequência e serão processadas como sequências com terminação nula.
- Se o parâmetro MAPPING-LEVEL estiver configurado como 1.2 ou superior e o parâmetro CHAR-VARYING estiver configurado como BINARY, as matrizes de caracteres serão mapeadas para uma sequência e serão processadas como dados binários.
- Se o parâmetro MAPPING-LEVEL estiver configurado como 1.2 ou superior e o parâmetro CHAR-VARYING estiver configurado como COLLAPSE, o espaço em branco final será ignorado para as sequências.
- A cláusula OCCURS DEPENDING ON é suportada em um nível de mapeamento de 4.0 ou superior. OCCURS DEPENDING ON complexo não é suportado. Isso significa que OCCURS DEPENDING ON é suportado somente para o último campo de uma estrutura.
- A cláusula OCCURS INDEXED BY é suportada em qualquer nível de mapeamento.

Descrição de Dados COBOL	Definição 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>)	

Descrição de Dados COBOL	Definição de esquema JSON
PIC S9 DISPLAY PIC S99 DISPLAY PIC S999 DISPLAY PIC S9999 DISPLAY PIC S9(n) DISPLAY PIC S9(n) COMP PIC S9(n) COMP-4 PIC S9(n) COMP-5 PIC S9(n) BINARY	<pre>"type": "integer", "minimum":- (n + 1), "maximum": n</pre> <p>em que n é o valor máximo que pode ser representado pelo padrão de '9' caracteres.</p>
PIC 9 DISPLAY PIC 99 DISPLAY PIC 999 DISPLAY PIC 9999 DISPLAY PIC 9(n) DISPLAY PIC 9(n) COMP PIC 9(n) COMP-4 PIC 9(n) COMP-5 PIC 9(n) BINARY	<pre>"type": "integer", "minimum":0, "maximum": n</pre> <p>em que n é o valor máximo que pode ser representado pelo padrão de '9' caracteres.</p>

Descrição de Dados COBOL	Definição 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>em que:</p> <p><i>x</i> é o valor mínimo que pode ser representado pelo padrão de '9' caracteres</p> <p><i>y</i> é o valor máximo que pode ser representado pelo padrão de '9' caracteres</p> <p><i>z</i> é a menor unidade disponível = 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>em que:</p> <p><i>y</i> é o valor máximo que pode ser representado pelo padrão de '9' caracteres</p> <p><i>z</i> é a menor unidade disponível = 1/10 "</p>
PIC S9(15) COMP-3 Suportado no nível de mapeamento 3.0 quando DATETIME=PACKED15	<pre>"type": "string", "format": "date-time"</pre> <p>O formato do registro de data e hora é definido pelo RFC3339.</p>
PIC S9(<i>m</i>)V9(<i>n</i>) DISPLAY Suportado no nível de mapeamento 1.2 e superior	<pre>"type": "number", "format": "decimal", "minimum": x , "maximum": y , "multipleOf": z</pre> <p>em que:</p> <p><i>x</i> é o valor mínimo que pode ser representado pelo padrão de '9' caracteres</p> <p><i>y</i> é o valor máximo que pode ser representado pelo padrão de '9' caracteres</p> <p><i>z</i> é a menor unidade disponível = 1/10 "</p>

Descrição de Dados COBOL	Definição de esquema JSON
<p>COMP-1</p> <p>Suportado no nível de mapeamento 1.2 e superior</p> <p>Nota: A representação de dados do IBM Hexadecimal Floating Point (HFP) não é exatamente igual à representação IEEE-754-1985 usada para JSON. Alguns valores podem não ser convertidos exatamente de uma representação para a outra.</p> <p>Alguns valores extremamente grandes ou pequenos podem não ser válidos para tipos de dados flutuantes. Alguns valores podem perder precisão quando convertidos para ou a partir de uma representação HFP. Se conversões precisas forem importantes, considere substituir o uso de tipos de dados COMP-1 por alternativas de precisão fixas.</p>	<pre>"type": "number", "format": "float"</pre>
<p>COMP-2</p> <p>Suportado no nível de mapeamento 1.2 e superior</p> <p>Nota: A representação de dados do IBM Hexadecimal Floating Point (HFP) não é exatamente igual à representação IEEE-754-1985 usada para JSON. Alguns valores podem não ser convertidos exatamente de uma representação para a outra.</p> <p>Alguns valores extremamente grandes ou pequenos podem não ser válidos para tipos de dados duplos. Alguns valores podem perder precisão quando convertidos para ou a partir de uma representação HFP. Se conversões precisas forem importantes, considere substituir o uso de tipos de dados COMP-2 por alternativas de precisão fixas .</p>	<pre>"type": "number", "format": "double"</pre>

Descrição de Dados COBOL	Definição de esquema JSON
<i>data description</i> OCCURS <i>n</i> TIMES	<p>No nível de mapeamento 4.0 e inferior</p> <p>Para primitivas:</p> <pre> "type": "array" "maxItems": n "minItems": n "items": { "tipo": "objeto", "properties": { name : { data description JSON } } "required": [name] } </pre> <p>Para estruturas:</p> <pre> "type": "array" "maxItems": n "minItems": n "items": { data description JSON } </pre> <p>Em que <i>data description JSON</i> é a representação de esquema JSON do COBOL <i>data description</i> e <i>name</i> é o nome do COBOL <i>data description</i>.</p> <p>No nível de mapeamento 4.1 e superior</p> <p>TRUNCATE-NULL-ARRAYS = DISABLED</p> <p>As matrizes estruturadas e primitivas são mapeadas conforme a seguir.</p> <pre> "type": "array" "maxItems": n "minItems": n "items": { data description JSON } </pre> <p>TRUNCATE-NULL-ARRAYS = ENABLED</p> <p>As matrizes primitivas são mapeadas conforme acima e as matrizes estruturadas conforme a seguir.</p> <pre> "type": "array" "maxItems": n "minItems": 0 "items": { data description JSON } </pre>

Descrição de Dados COBOL	Definição de esquema JSON
<i>data description</i> OCCURS <i>n</i> TO <i>m</i> TIMES DEPENDING ON <i>t</i> Suportado no nível de mapeamento 4.0	<pre>"field-name" : { "type": "array" , "maxItems": <i>m</i> "minItems": <i>n</i> "items": { ... } }</pre> O conteúdo do item de matriz depende do tipo de dados usado.
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	Quando CHAR-OCCURS =STRING: <pre>"field-name" : { "type": "string" , "maxLength": <i>n</i> }</pre> Ele é uma sequência.

Descrição de Dados COBOL	Definição de esquema JSON
	<p>Quando CHAR-OCCURS =ARRAY:</p> <p>No nível de mapeamento 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>Isto é uma matriz de caracteres únicos.</p> <p>No nível de mapeamento 4.1 e superior</p> <p>TRUNCATE-NULL-ARRAYS = DISABLED</p> <p>As matrizes estruturadas e primitivas são mapeadas conforme a seguir.</p> <pre> "type":"array" "maxItems": n "minItems": n "items":{ data description JSON } </pre> <p>TRUNCATE-NULL-ARRAYS = ENABLED</p> <p>As matrizes primitivas são mapeadas conforme acima e as matrizes estruturadas conforme a seguir.</p> <pre> "type":"array" "maxItems": n "minItems": 0 "items":{ data description JSON } </pre>

Descrição de Dados COBOL	Definição de esquema JSON
<pre> PIC X OCCURS n TO m TIMES DEPENDING ON t PIC A OCCURS n TO m TIMES DEPENDING ON t PIC G DISPLAY-1 OCCURS n TO m TIMES DEPENDING ON t PIC N OCCURS n TO m TIMES DEPENDING ON t </pre>	<p>Quando CHAR-OCCURS =STRING:</p> <pre> "field-name" : { "type": "string" , "maxLength": m "minLength": n } </pre>
<p>PIC N(<i>n</i>) USAGE NATIONAL</p> <p>Quando CHAR-USAGE =NATIONAL : PIC N(<i>n</i>)</p>	<p>No nível de mapeamento 4.0 e superior:</p> <pre> "type": "string", "maxLength": n </pre> <p>No tempo de execução, CICS preenche o campo de estrutura de dados do aplicativo com dados UTF-16.</p>

Esquema JSON para mapeamento COBOL:

O programa utilitário DFHJS2LS suporta mapeamentos entre o esquema JSON e as estruturas de dados COBOL.

Os assistentes do CICS geram nomes de campo válidos e exclusivos para variáveis COBOL a partir dos nomes de elemento de esquema usando as regras a seguir:

1. Palavras reservadas COBOL são prefixadas com ' X' inicial.
Por exemplo, DISPLAY se torna XDISPLAY.
2. Caracteres diferentes de A-Z, a-z, 0-9 ou hífen são substituídos por ' X' inicial.
Por exemplo, monthly_total se torna monthlyXtotal.
3. Se o último caractere for um hífen, ele será substituído por ' X' inicial.
Por exemplo, ca-request- se torna ca-requestX.

4. Nomes duplicados no mesmo escopo se tornam exclusivos pela adição de um ou dois dígitos numéricos na segunda instância e nas instâncias subsequentes do nome.
Por exemplo, três instâncias de year se tornam year , year1 e year2.
5. Um esquema JSON especifica que uma variável possui variação de cardinalidade se possui um valor de "type" igual a "array" e as palavras-chave "minItems" e "maxItems" foram omitidas ou possuem valores diferentes. Se o esquema especifica que a variável possui variação de cardinalidade, os nomes de campo são criados com sufixos "_cont" e "_num".
Para obter mais informações, consulte “Matrizes variáveis de elementos no DFHJS2LS” na página 282.
6. Um esquema JSON especifica que uma variável é opcional se ela não aparece na matriz de palavra-chave "required" que está associada ao tipo de "object" de esquema JSON de inclusão. Para campos opcionais, um campo adicional é gerado com um sufixo _num incluído no nome de elemento. No tempo de execução, ele é zero para indicar que o valor estava ausente nos dados JSON e não zero se o valor estava presente nos dados JSON.
7. Os nomes de campo são limitados a 28 caracteres. Se um nome gerado, incluindo o prefixo e o sufixo, exceder esse comprimento, o nome de elemento será truncado.

DFHJS2LS mapeia tipos de esquema para elementos de descrição de dados COBOL usando o nível de mapeamento especificado de acordo com a tabela a seguir. Observe os pontos a seguir:

- Se o parâmetro **MAPPING-LEVEL** for configurado como 1.2 ou superior e o parâmetro **CHAR-VARYING** for configurado como NULL, os dados de caractere de comprimento variável serão mapeados para sequências com terminação nula e um caractere extra será alocado para o terminador NULL.
- Se o parâmetro **MAPPING-LEVEL** for configurado como 1.2 ou superior e o parâmetro **CHAR-VARYING** for configurado como YES, os dados de caractere de comprimento variável serão mapeados para dois elementos relacionados: um campo de comprimento e um campo de dados. Por exemplo:

```
"textString": {
  "type": "string",
  "maxLength": 10000,
  "minLength": 1
}
```

mapeia para:

```
15 textString-length PIC S9999 COMP-5 SYNC
15 textString PIC X(10000)
```

Palavra-chave do Esquema JSON	Descrição de Dados COBOL
Todos de: "type": "null" "type": [] "enum": [] "allOf" "anyOf" "noneOf" "not"	Não-suportado

Palavra-chave do Esquema JSON	Descrição de Dados COBOL
"\$schema": "http://json-schema.org/draft-04/schema#"	Esta palavra-chave é ignorada, mas assume-se que ela é compatível com a especificação de Esquema JSON 04 de rascunho.
"title": "same text" "description": "more text"	Essas palavras-chave são ignoradas.
"format": "<predefined values>"	A palavra-chave "format" é utilizada para modificar a estrutura gerada ou o valor de tempo de execução. Consulte as informações posteriormente nesta tabela para o uso suportado de "format".
"type": "array", "items": {<JSON Sub-schema>}, "additionalItems": false, "maxItems": m , "minItems": n	<p>A única forma de matriz JSON suportada atualmente é um número repetido dos mesmos valores de tipo. O <JSON Sub-schema> deve definir um "type" suportado, mas esse "type" não pode ser "array". Esta é uma restrição na estrutura de linguagem gerada.</p> <p>"additionalItems" é assumido como false e nenhum outro valor é suportado.</p> <p>Se "minItems" e "maxItems" estiverem presentes e forem iguais, a matriz será tratada como cardinalidade fixa, caso contrário, será tratada como cardinalidade variável. Consulte "Matrizes variáveis de elementos no DFHJS2LS" na página 282.</p>
"type": "array", "uniqueItems": true	"uniqueItems" não é suportado com matrizes JSON.
"type": "object", "additionalProperties": false, "properties": { ["<element name>": {<JSON Sub-schema> } [,]]* } "required": [["<element name>" [,]]*]	<p>O único formato do objeto JSON que é suportado atualmente é um conjunto fixo de elementos nomeados.</p> <p>Isto gera uma estrutura (ou subestrutura) que usa os nomes de elemento.</p> <p>"additionalProperties" é assumido como false e nenhum outro valor é suportado.</p> <p>Qualquer elemento no objeto "properties" é considerado "optional" se ele não está na matriz "required" ou se nenhuma matriz "required" existe. Um elemento "optional" recebe uma cardinalidade variável de zero a X; em que X é 1 ou o número máximo de itens na matriz no qual esse item é definido como uma matriz. Consulte "Matrizes variáveis de elementos no DFHJS2LS" na página 282.</p>

Palavra-chave do Esquema JSON	Descrição de Dados COBOL
<pre>"type": "object", "maxProperties": m, "minProperties": n, "patternProperties": {}, "dependencies":</pre>	Nenhuma dessas palavras-chave é suportada com objetos JSON.
<pre>"type": "string" "maxLength": m "pattern": "regular expression>", "minLength": l</pre>	<p>PIC X(z)</p> <p>em que o valor de <i>z</i> é baseado em <i>m</i>, porém é dependente das configurações do parâmetro CHAR-VARYING.</p> <p><i>m</i> é baseado na palavra-chave "maxLength" e tratado como uma sequência de comprimento fixo.</p> <p>As restrições "pattern" e "minLength" são transmitidas para a estrutura de linguagem somente como um comentário.</p>
<pre>"type": "string" "maxLength": m</pre>	<p>Quando CCSID=1200 no nível de mapeamento 4.0 e superior:</p> <p>PIC N(z) USAGE NATIONAL</p> <p>em que o valor de <i>z</i> é baseado em <i>m</i>, porém é dependente das configurações do parâmetro CHAR-VARYING.</p> <p><i>m</i> é baseado na palavra-chave "maxLength" e tratado como uma sequência de comprimento fixo.</p>
<pre>"*name*":{ "type": "string", "format": "date-time" }</pre>	<p>PIC S9(15) COMP-3</p> <p>Todos suportados quando DATETIME=PACKED15</p> <p>Observe que "maxLength" e "minLength" não são suportados para este formato</p>
<pre>"*name*":{ "type": "string", "format": "uri" }</pre>	<p>PIC X(m)</p> <p>em que <i>m</i> é baseado na palavra-chave "maxLength" e tratado como uma sequência de comprimento fixo.</p> <p>Quando CCSID=1200 no nível de mapeamento 4.0 e superior:</p> <p>PIC N(m) USAGE NATIONAL</p> <p>em que <i>m</i> é baseado na palavra-chave "maxLength" e tratado como uma sequência de comprimento fixo.</p>
<pre>"*name*":{ "type": "string", "format": "base64Binary" }</pre>	<p>PIC X(m)</p> <p>em que <i>m</i> é baseado na palavra-chave "maxLength"</p>

Palavra-chave do Esquema JSON	Descrição de Dados COBOL
<pre>"*name*":{ "type": "string", "format":"hexBinary" }</pre>	<p>PIC X(<i>m</i>)</p> <p>em que <i>m</i> é baseado na palavra-chave "maxLength"</p>
<pre>"*name*":{ "type": "string", "format":"<predefined>" }</pre>	<p>PIC X(<i>m</i>)</p> <p>em que <i>m</i> é baseado na palavra-chave "maxLength" e tratado como sequência de comprimento fixo e em que <predefined> é um de: <i>e-mail</i>, <i>hostname</i>, <i>ipv4</i> ou <i>ipv6</i>. Um "padrão" relevante é usado e transmitido ao comentário.</p> <p>Quando CCSID=1200 no nível de mapeamento 4.0 e superior:</p> <p>PIC N(<i>m</i>) USAGE NATIONAL</p> <p>em que <i>m</i> é baseado na palavra-chave "maxLength" e tratado como sequência de comprimento fixo e em que <predefined> é um de: <i>e-mail</i>, <i>hostname</i>, <i>ipv4</i> ou <i>ipv6</i>. Um "padrão" relevante é usado e transmitido ao comentário.</p>
<pre>"type":"boolean"</pre>	<p>PIC X DISPLAY</p> <p>O valor x'00' significa false, x'01' significa true.</p>
<pre>"type": "integer", "maxExclusive": true, "minExclusive": true, "multipleOf": n</pre>	<p>As restrições "maxExclusive" e "minExclusive" são transmitidas à estrutura de linguagem somente como um comentário.</p> <p>"multipleOf" é ignorado.</p>
<pre>"type"="integer", minimum=0, maximum=255</pre>	<p>Nível de mapeamento 3.0 e abaixo:</p> <p>PIC X DISPLAY</p> <p>Nível de mapeamento 4.0 e acima (ou quando o parâmetro INTEGER-AS-PIC9 tiver sido especificado, independentemente do nível de mapeamento):</p> <p>PIC 9(<i>z</i>) COMP-5 SYNC</p> <p>ou</p> <p>PIC 9(<i>z</i>) DISPLAY</p> <p>em que $10^{(z-1)} < m \leq 10^z$</p>

Palavra-chave do Esquema JSON	Descrição de Dados COBOL
"type": "integer", minimum:-128, maximum:127	Nível de mapeamento 3.0 e abaixo: PIC X DISPLAY Nível de mapeamento 4.0 e acima (ou quando o parâmetro INTEGER-AS-PIC9 tiver sido especificado, independentemente do nível de mapeamento): PIC S9(z) COMP-5 SYNC ou PIC S9(z) DISPLAY em que $10^{(z-1)} < m \leq 10^z$
"type": "integer", minimum:0, maximum; m	PIC 9(z) COMP-5 SYNC ou PIC 9(z) DISPLAY em que $10^{(z-1)} < m \leq 10^z$
"type": "integer", mínimo:- m , máximo: m -1	PIC S9(z) COMP-5 SYNC ou PIC S9(z) DISPLAY em que $10^{(z-1)} < m \leq 10^z$
"type": "number", "maximum": m, "minimum": n, "maxExclusive": true, "minExclusive": true, "multipleOf": n	As restrições "maximum" , "minimum" , "maxExclusive" e "minExclusive" são transmitidas à estrutura de linguagem somente como um comentário. "multipleOf" é ignorado.
"type": "number" "format": "decimal"	PIC 9(p)V9(n) COMP-3 em que <i>p</i> e <i>n</i> são valores padrão.

Palavra-chave do Esquema JSON	Descrição de Dados COBOL
<pre>"type": "number" "format": "float"</pre>	<p>Nível de mapeamento 1.1 e inferior:</p> <ul style="list-style-type: none"> PIC X(32) <p>Nível de mapeamento 1.2 e posterior:</p> <ul style="list-style-type: none"> COMP-1 <p>Nota: A representação de dados do IBM Hexadecimal Floating Point (HFP) não é exatamente igual à representação IEEE-754-1985 usada para JSON. Alguns valores podem não ser convertidos exatamente de uma representação para a outra.</p> <p>Alguns valores extremamente grandes ou pequenos podem não ser válidos para tipos de dados flutuantes. Alguns valores podem perder precisão quando convertidos para ou a partir de uma representação HFP. Se conversões precisas forem importantes, considere substituir o uso de tipos de dados COMP-1 por alternativas de precisão fixas.</p>
<pre>"type": "number" "format": "double"</pre>	<p>Nível de mapeamento 1.1 e inferior:</p> <ul style="list-style-type: none"> PIC X(32) <p>Nível de mapeamento 1.2 e posterior:</p> <ul style="list-style-type: none"> COMP-2 <p>Nota: A representação de dados do IBM Hexadecimal Floating Point (HFP) não é exatamente igual à representação IEEE-754-1985 usada para JSON. Alguns valores podem não ser convertidos exatamente de uma representação para a outra.</p> <p>Alguns valores extremamente grandes ou pequenos podem não ser válidos para tipos de dados duplos. Alguns valores podem perder precisão quando convertidos para ou a partir de uma representação HFP. Se conversões precisas forem importantes, considere substituir o uso de tipos de dados COMP-2 por alternativas de precisão fixas .</p>

Nota: O CICS não pode transformar valores de número inteiro maiores do que o valor máximo para um longo sinalizado ($2^{63} - 1$), a menos que eles sejam colocados entre aspas.

Nota: Os valores mínimo e máximo especificados no esquema para tipos numéricos são utilizados somente para mapear para um tipo de dados COBOL. Os dados não são validados com relação a esses valores no tempo de execução.

Alguns dos tipos de esquema que são mostrados na tabela mapeiam para um formato COBOL de COMP-5 SYNC ou de DISPLAY, dependendo dos valores (se houver) que são especificados nas palavras-chave `minimum` e `maximum`:

- Para tipos sinalizados (`short`, `int`, e `long`), `DISPLAY` é usado quando o seguinte é especificado:

```
"maximum":  
  a  
"minimum":  
  -a
```

em que *a* é uma sequência de '9'.

- Para tipos não sinalizados (`unsignedShort`, `unsignedInt` e `unsignedLong`), `DISPLAY` é usado quando o seguinte é especificado:

```
"maximum":  
  a  
"minimum":0
```

em que *a* é uma sequência de '9'.

Quando qualquer outro valor é especificado, ou nenhum valor é especificado, `COMP-5 SYNC` é usado.

C e C++ para mapeamento de esquema JSON:

O programa utilitário `DFHLS2JS` suporta mapeamentos entre tipos de dados C e C++ e definições de esquema JSON.

Os nomes C e C++ são convertidos em nomes JSON de acordo com as seguintes regras:

1. Os caracteres que não são válidos em nomes de propriedades JSON são substituídos por 'X'.
Por exemplo, `monthly-total` se torna `monthlyXtotal`.
2. Nomes duplicados se tornam exclusivos pela adição de um ou mais dígitos numéricos.
Por exemplo, duas instâncias de `year` se tornam `year` e `year1`.

O `DFHLS2JS` mapeia tipos de dados C e C++ para os elementos de esquema de acordo com a tabela a seguir. Os tipos C e C++ que não são mostrados na tabela não são suportados pelo `DFHLS2JS`. O qualificador `_Packed` é suportado para estruturas. Estas restrições se aplicam:

- Os arquivos de cabeçalho devem conter uma instância `struct` de nível superior.
- Não é possível declarar um tipo de estrutura que contenha ele mesmo como um membro.
- Os seguintes tipos de dados C e C++ não são suportados:
 - `decimal`
 - `long double`
 - `wchar_t` (somente C++)
- Os seguintes são ignorados se estão presentes no arquivo de cabeçalho.

Especificadores de classe de armazenamento:

`auto`

register
static
extern
mutable

Qualificadores

const
volatile
_Export (somente C++)

Especificações de funções

inline (somente C++)
virtual (somente C++)

Valores iniciais

- O arquivo de cabeçalho não deve conter estes itens:
 - Uniãos
 - Declarações de classe
 - Tipos de dados de enumeração
 - Variáveis de tipo de ponteiro
 - Declarações de modelo
 - Macros predefinidas; isto é, macros com nomes que iniciam e terminam com dois caracteres de sublinhado (__)
 - A sequência de continuação de linha (um símbolo \ que é imediatamente seguido por um caractere de nova linha)
 - Declaradores de função de protótipo
 - Diretivas de pré-processadores
 - Campos de bits
 - A palavra-chave __cdecl (ou _cdecl) (somente C++)
- O programador do aplicativo deve utilizar um compilador de 32 bits para assegurar que um int seja mapeado para 4 bytes.
- As palavras-chave reservadas por C++ a seguir não são suportadas:
 - explicit
 - using
 - namespace
 - typename
 - typeid
- Se o parâmetro **MAPPING-LEVEL** estiver configurado como 1.2 ou superior e o parâmetro **CHAR-VARYING** estiver configurado como NULL, as matrizes de caracteres serão mapeadas para um string e serão processadas como sequências com terminação nula.
- Se o parâmetro **MAPPING-LEVEL** for configurado como 1.2 ou superior e o parâmetro **CHAR-VARYING** for configurado como BINARY, as matrizes de caracteres serão mapeadas para xsd:base64Binary e serão processadas como dados binários.
- Se o parâmetro **MAPPING-LEVEL** for configurado como 1.2 ou superior e o parâmetro **CHAR-VARYING** for configurado como COLLAPSE, <xsd:whiteSpace value="collapse"/> será gerado para sequências.

Tipo de Dados C e C++	simpleType de Esquema
char[z]	"type":"string" "maxLength": z

Tipo de Dados C e C++	simpleType de Esquema
char16_t[<i>n</i>]	<p>No nível de mapeamento 4.0 e superior:</p> <pre>"type": "string" "maxLength": <i>n</i></pre> <p>No tempo de execução, CICS preenche o campo de estrutura de dados do aplicativo com dados UTF-16.</p>
char[8] Suportado no nível de mapeamento 3.0 e superior quando DATETIME=PACKED15	<pre>"type": "string" "format": "date-time"</pre> <p>O formato do registro de data e hora é definido pelo RFC3339.</p>
char short int long long long	<pre>"type": "integer", "minimum": - (<i>n</i> + 1), "maximum": <i>n</i></pre> <p>em que <i>n</i> é o valor máximo que pode ser representado pelo primitivo.</p>
unsigned char unsigned short unsigned int unsigned long unsigned long long	<pre>"type": "integer", "minimum": 0, "maximum": <i>n</i></pre> <p>em que <i>n</i> é o valor máximo que pode ser representado pelo primitivo.</p>
bool (somente C++)	<pre>"type": "boolean"</pre>
float Suportado no nível de mapeamento 1.2 e superior.	<pre>"type": "number", "format": "float"</pre> <p>Nota: A representação de dados do IBM Hexadecimal Floating Point (HFP) não é exatamente igual à representação IEEE-754-1985 usada para JSON. Alguns valores podem não ser convertidos exatamente de uma representação para a outra. Alguns valores extremamente grandes ou pequenos podem não ser válidos para tipos de dados flutuantes. Alguns valores podem perder a precisão quando convertidos para ou a partir de uma representação HFP. Se conversões precisas forem importantes, considere substituir o uso de tipos de dados flutuantes por alternativas de precisão fixas.</p>
duplo Suportado no nível de mapeamento 1.2 e superior.	<pre>"type": "number", "format": "double"</pre> <p>Nota: A representação de dados do IBM Hexadecimal Floating Point (HFP) não é exatamente igual à representação IEEE-754-1985 usada para JSON. Alguns valores podem não ser convertidos exatamente de uma representação para a outra. Alguns valores extremamente grandes ou pequenos podem não ser válidos para tipos de dados duplo. Alguns valores podem perder a precisão quando convertidos para ou a partir de uma representação HFP. Se conversões precisas forem importantes, considere substituir o uso de tipos duplos de dados por alternativas de precisão fixas.</p>

Tipo de Dados C e C++	simpleType de Esquema
<i>type name [n]</i>	<p>Para primitivas:</p> <pre> "type": "array" "maxItems": n "minItems": n "items": { "tipo": "objeto", "properties": { name : { type JSON } } "required": [name] } </pre> <p>Para estruturas:</p> <pre> "type": "array" "maxItems": n "minItems": n "items": { type JSON } </pre> <p>Em que <i>type JSON</i> é a representação de esquema JSON do tipo C ou C++.</p>

Esquema JSON para mapeamento C e C++:

Os programas utilitários DFHJS2LS suportam mapeamentos entre os esquemas JSON e os tipos de dados C e C++.

Os assistentes do CICS geram nomes de campo válidos e exclusivos para variáveis C e C++ a partir dos nomes de elemento de esquema usando as regras a seguir:

1. Caracteres diferentes de A-Z, a-z, 0-9 ou _ são substituídos por ' X' inicial.
Por exemplo, *monthly-total* se torna *monthlyXtotal*.
2. Se o primeiro caractere não for um caractere alfabético, ele será substituído por um ' X' inicial.
Por exemplo, *_monthlysummary* se torna *Xmonthlysummary*.
3. Nomes duplicados no mesmo escopo se tornam exclusivos pela adição de um ou dois dígitos numéricos na segunda instância e nas instâncias subsequentes do nome.
Por exemplo, três instâncias de *year* se tornam *year* , *year1* e *year2*.
4. Um esquema JSON especifica que uma variável possui variação de cardinalidade se possui um valor de "type" igual a "array" e as palavras-chave "minItems" e "maxItems" foram omitidas ou possuem valores diferentes. Se o esquema especifica que a variável possui variação de cardinalidade, os nomes de campo são criados com sufixos "_cont" e "_num".
Para obter mais informações, consulte “Matrizes variáveis de elementos no DFHJS2LS” na página 282.

5. Um esquema JSON especifica que uma variável é opcional se ela não aparece na matriz de palavra-chave "required" que está associada ao tipo de "object" de esquema JSON de inclusão. Para campos opcionais, um campo adicional é gerado com um sufixo _num incluído no nome de elemento. No tempo de execução, ele é zero para indicar que o valor estava ausente nos dados JSON e não zero se o valor estava presente nos dados JSON.
6. Os nomes de campo são limitados a 50 caracteres. Se um nome gerado, incluindo qualquer prefixo e sufixo, exceder esse comprimento, o nome de elemento será truncado.

O DFHJS2LS mapeia os valores de tipo de esquema JSON para tipos de dados C e C++ de acordo com a tabela a seguir. As regras a seguir também se aplicam:

- Se o parâmetro **MAPPING-LEVEL** for configurado como 1.2 ou superior e o parâmetro **CHAR-VARYING** for configurado como NULL, os dados de caractere de comprimento variável serão mapeados para sequências com terminação nula e um caractere extra será alocado para o terminador NULL.
- Se o parâmetro **MAPPING-LEVEL** for configurado como 1.2 ou superior e o parâmetro **CHAR-VARYING** for configurado como YES, os dados de caractere de comprimento variável serão mapeados para dois elementos relacionados: um campo de comprimento e um campo de dados.

Palavra-chave do Esquema JSON	Tipo de Dados C e C++
Todos de: "type": "null" "type": [] "enum": [] "allOf" "anyOf" "noneOf" "not"	Não-suportado
"\$schema": "http://json-schema.org/draft-04/schema#"	Esta palavra-chave é ignorada, mas assume-se que ela é compatível com a especificação de Esquema JSON 04 de rascunho.
"title": "same text" "description": "more text"	Essas palavras-chave são ignoradas.
"format": "<predefined values>"	A palavra-chave "format" é utilizada para modificar a estrutura gerada ou o valor de tempo de execução. Consulte as informações a seguir para o uso suportado de "format".

Palavra-chave do Esquema JSON	Tipo de Dados C e C++
<pre>"type": "array", "items": {<JSON Sub-schema>}, "additionalItems": false, "maxItems": m , "minItems": n</pre>	<p>A única forma de matriz JSON suportada atualmente é um número repetido dos mesmos valores de tipo. O <JSON Sub-schema> deve definir um "type" suportado , mas esse "type" não pode ser "array". Esta é uma restrição na estrutura de linguagem gerada.</p> <p>"additionalItems" é assumido como false e nenhum outro valor é suportado.</p> <p>Se "minItems" e "maxItems" estiverem presentes e forem iguais, a matriz será tratada como cardinalidade fixa, caso contrário, será tratada como cardinalidade variável. Consulte “Matrizes variáveis de elementos no DFHJS2LS” na página 282.</p>
<pre>"type": "array", "uniqueItems": true</pre>	<p>"uniqueItems" não é suportado com matrizes JSON. O <JSON Sub-schema> deve definir um "type" suportado , mas esse "type" não pode ser "array" . Esta é uma restrição na estrutura de linguagem gerada.</p>
<pre>"type": "object", "additionalProperties": false, "properties": { ["<element name>": {<JSON Sub-schema>} [,]]* } "required": [["<element name>" [,]]*]</pre>	<p>A única forma de objeto JSON atualmente suportada é um conjunto fixo de elementos nomeados.</p> <p>Isso gerará uma estrutura (ou subestrutura) usando os nomes de elementos.</p> <p>"additionalProperties" é assumido como false e nenhum outro valor é suportado.</p> <p>Qualquer elemento no objeto "properties" é considerado "optional" se ele não está na matriz "required" ou se nenhuma matriz "required" existe. Um elemento "optional" recebe uma cardinalidade variável de zero a X; em que X é 1 ou o número máximo de itens na matriz no qual esse item é definido como uma matriz. Consulte “Matrizes variáveis de elementos no DFHJS2LS” na página 282.</p>
<pre>"type": "object", "maxProperties": m, "minProperties": n, "patternProperties": {}, "dependencies":</pre>	<p>Nenhuma dessas palavras-chave é suportada com objetos JSON.</p>

Palavra-chave do Esquema JSON	Tipo de Dados C e C++
<pre>"type": "string" "maxLength": m "pattern": "regular expression>", "minLength": l</pre>	<p>char[z]</p> <p>em que o valor de z é baseado em m, porém é dependente das configurações do parâmetro CHAR-VARYING.</p> <p>m é baseado na palavra-chave "maxLength" e tratado como uma sequência de comprimento fixo.</p> <p>As restrições "pattern" e "minLength" são transmitidas para a estrutura de linguagem somente como um comentário.</p>
<pre>"type": "string" "maxLength": m</pre>	<p>Quando CCSID=1200 no nível de mapeamento 4.0 e superior:</p> <p>char16_t[z]</p> <p>em que o valor de z é baseado em m, porém é dependente das configurações do parâmetro CHAR-VARYING.</p> <p>m é baseado na palavra-chave "maxLength" e tratado como uma sequência de comprimento fixo.</p>
<pre>"*name*":{ "type": "string", "format": "date-time" }</pre>	<p>char[8]</p> <p>Todos suportados quando DATETIME=PACKED15</p>
<pre>"*name*":{ "type": "string", "format": "uri" }</pre>	<p>char[m]</p> <p>em que m é baseado na palavra-chave "maxLength" e tratado como uma sequência de comprimento fixo.</p> <p>Quando CCSID=1200 no nível de mapeamento 4.0 e superior:</p> <p>char16_t[m]</p> <p>em que m é baseado na palavra-chave "maxLength" e tratado como uma sequência de comprimento fixo.</p>
<pre>"*name*":{ "type": "string", "format": "base64Binary" }</pre>	<p>char[m]</p> <p>em que m é baseado na palavra-chave "maxLength".</p>
<pre>"*name*":{ "type": "string", "format": "hexBinary" }</pre>	<p>char[m]</p> <p>em que m é baseado na palavra-chave "maxLength".</p>

Palavra-chave do Esquema JSON	Tipo de Dados C e C++
<pre>"*name*":{ "type": "string", "format": "<predefined>" }</pre>	<p>char[<i>m</i>]</p> <p>em que <i>m</i> é baseado na palavra-chave "maxLength" e tratado como sequência de comprimento fixo e em que <predefined> é um de: <i>email</i> , <i>hostname</i> , <i>ipv4</i> ou <i>ipv6</i>. Uma "pattern" relevante é usado e transmitido ao comentário.</p> <p>Quando CCSID=1200 no nível de mapeamento 4.0 e superior:</p> <p>char16_t[<i>m</i>]</p> <p>em que <i>m</i> é baseado na palavra-chave "maxLength" e tratado como sequência de comprimento fixo e em que <predefined> é um de: <i>e-mail</i> , <i>hostname</i> , <i>ipv4</i> ou <i>ipv6</i> . Um "padrão" relevante é usado e transmitido ao comentário.</p>
"type": "boolean"	<p>bool (somente C++)</p> <p>short (somente C)</p>
<pre>"type": "integer", "maxExclusive": true, "minExclusive": true, "multipleOf": n</pre>	<p>As restrições "maxExclusive" e "minExclusive" são transmitidas à estrutura de linguagem somente como um comentário.</p> <p>"multipleOf" é ignorado.</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

Palavra-chave do Esquema JSON	Tipo de Dados C e C++
"type": "integer", minimum:0, maximum:18446744073709551615	unsigned long long
"type": "number", "maximum": m, "minimum": n, "maxExclusive": true, "minExclusive": true, "multipleOf": n	As restrições "maximum", "minimum", "maxExclusive" e "minExclusive" são transmitidas à estrutura de linguagem somente como um comentário. "multipleOf" é ignorado.
"type": "number" "format": "float"	Nível de mapeamento 1.1 e inferior: <ul style="list-style-type: none"> char[32] Nível de mapeamento 1.2 e posterior: <ul style="list-style-type: none"> float(*) <p>Nota: A representação de dados do IBM Hexadecimal Floating Point (HFP) não é exatamente igual à representação IEEE-754-1985 usada para JSON. Alguns valores podem não ser convertidos exatamente de uma representação para a outra. Alguns valores extremamente grandes ou pequenos podem não ser válidos para tipos de dados flutuantes. Alguns valores podem perder a precisão quando convertidos para ou a partir de uma representação HFP. Se conversões precisas forem importantes, considere substituir o uso de tipos de dados flutuantes por alternativas de precisão fixas.</p>
"type": "number" "format": "double"	Nível de mapeamento 1.0 e inferior: <ul style="list-style-type: none"> char[32] Nível de mapeamento 1.2 e posterior: <ul style="list-style-type: none"> double(*) <p>Nota: A representação de dados do IBM Hexadecimal Floating Point (HFP) não é exatamente igual à representação IEEE-754-1985 usada para JSON. Alguns valores podem não ser convertidos exatamente de uma representação para a outra. Alguns valores extremamente grandes ou pequenos podem não ser válidos para tipos de dados duplo. Alguns valores podem perder a precisão quando convertidos para ou a partir de uma representação HFP. Se conversões precisas forem importantes, considere substituir o uso de tipos duplos de dados por alternativas de precisão fixas.</p>

Nota: O CICS não pode transformar valores de número inteiro maiores que o valor máximo para um longo sinalizado ($2^{63} - 1$) a menos que eles sejam colocados entre aspas.

Nota: Os valores mínimo e máximo especificados no esquema para tipos numéricos são usados somente para mapear para um tipo de dados C ou C++. Os dados não são validados com relação a esses valores no tempo de execução.

Mapeamento de esquema PL/I para JSON:

O programa utilitário DFHLS2JS suporta mapeamentos entre as estruturas de dados PL/I e as definições de esquema JSON. Como o compilador Enterprise PL/I e os compiladores PL/I mais antigos diferem, duas opções de linguagem são suportadas: PLI-ENTERPRISE e PLI-OTHER.

Os nomes PL/I são convertidos em nomes JSON de acordo com as regras a seguir:

1. Os caracteres que não são válidos em nomes de propriedades JSON são substituídos por ' x '.
Por exemplo, `monthly$total` se torna `monthlyxtotal`.
2. Nomes duplicados se tornam exclusivos pela adição de um ou mais dígitos numéricos.
Por exemplo, duas instâncias de `year` se tornam `year` e `year1`.

O DFHLS2JS mapeia tipos de dados PL/I para elementos de esquema, de acordo com a tabela a seguir. Os tipos PL/I que não são mostrados na tabela não são suportados pelo DFHLS2JS. As restrições a seguir também se aplicam:

- Os itens de dados com o atributo COMPLEX não são suportados.
- Os itens de dados com o atributo FLOAT são suportados em um nível de mapeamento 1.2 ou superior. Enterprise PL/I FLOAT IEEE não é suportado.
- As sequências DBCS puras VARYING e VARYINGZ são suportadas em um nível de mapeamento 1.2 ou superior.
- Os itens de dados que são especificados como DECIMAL(*p* , *q*) são suportados somente quando $p \geq q$
- Os itens de dados que são especificados como BINARY(*p* , *q*) são suportados somente quando $q = 0$.
- Se o atributo PRECISION for especificado para um item de dados, ele será ignorado.
- As sequências PICTURE não são suportadas.
- Itens de dados ORDINAL são tratados como tipos de dados FIXED BINARY(7) .
- Se o parâmetro **MAPPING-LEVEL** estiver configurado como 1.2 ou superior e o parâmetro **CHAR-VARYING** estiver configurado como NULL, as matrizes de caracteres serão mapeadas para um string e serão processadas como sequências com terminação nula; este mapeamento não se aplica para Enterprise PL/I.
- Se o parâmetro **MAPPING-LEVEL** estiver configurado como 1.2 ou superior e o parâmetro **CHAR-VARYING** estiver configurado como BINARY, as matrizes de caracteres serão mapeadas para string e serão processadas como dados binários.
- Se o parâmetro **MAPPING-LEVEL** estiver configurado como 1.2 ou superior e o parâmetro **CHAR-VARYING** estiver configurado como COLLAPSE, os espaços em branco inicial e final serão removidos e vários espaços serão substituídos por um único espaço.

O DFHLS2JS não implementa completamente os algoritmos de preenchimento de PL/I; portanto, você deve declarar os bytes de preenchimento explicitamente em sua estrutura de dados. O DFHLS2JS emite uma mensagem se detecta que bytes de preenchimento estão ausentes. Cada estrutura de nível superior deve iniciar em um limite de palavra dupla e cada byte na estrutura deve ser mapeado para o limite correto. Considere este fragmento de código:

```
3 FIELD1 FIXED BINARY(7),
3 FIELD2 FIXED BINARY(31),
3 FIELD3 FIXED BINARY(63);
```

Neste exemplo:

- FIELD1 possui 1 byte de comprimento e pode ser alinhado em qualquer limite.
- FIELD2 tem 4 bytes de comprimento e deve ser alinhado em um limite de palavra inteira.
- FIELD3 tem 8 bytes de comprimento e deve ser alinhado em um limite de palavra dupla.

O compilador Enterprise PL/I alinha os campos na ordem a seguir:

1. FIELD3 é alinhado primeiro porque ele tem os requisitos de limite mais fortes.
2. FIELD2 é alinhado no limite de palavra inteira imediatamente antes de FIELD3.
3. FIELD1 é alinhado no limite de byte imediatamente antes de FIELD3.

Finalmente, para que a estrutura inteira esteja alinhada em um limite de palavra inteira, o compilador insere três bytes de preenchimento imediatamente antes de FIELD1.

Como DFHLS2JS não insere bytes de preenchimento equivalentes, você deve declará-los explicitamente antes da estrutura ser processada por DFHLS2JS. Por exemplo:

```
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);
```

Como alternativa, é possível mudar a estrutura para declarar todos os campos como desalinhados e recompilar o aplicativo que usa a estrutura. Para obter mais informações sobre requisitos de alinhamento de memória estrutural PL/I, consulte *Referência de Linguagem Enterprise PL/I*.

Descrição de Dados PL/I	Definição de esquema JSON
FIXED BINARY (<i>n</i>)	<pre>"type": "integer", "minimum":- (<i>n</i> + 1), "maximum": <i>n</i></pre> <p>em que <i>n</i> é o valor máximo que pode ser representado pelo primitivo.</p>
UNSIGNED FIXED BINARY(<i>n</i>) Restrição: Somente PL/I c Corporativo	<pre>"type": "integer", "minimum":0, "maximum": <i>n</i></pre> <p>em que <i>n</i> é o valor máximo que pode ser representado pelo primitivo.</p>

Descrição de Dados PL/I	Definição de esquema JSON
FIXED DECIMAL(<i>n</i> , <i>m</i>)	<pre>"type":"number", "format":"decimal", "multipleOf": x , "maximum": y , "mínimo":- z</pre> <p>em que:</p> <p><i>x</i> é a menor unidade disponível = $1/10^m$</p> <p><i>y</i> é o valor máximo que pode ser representado pela combinação de <i>n</i> e <i>m</i></p> <p><i>z</i> é o valor máximo que pode ser representado pela combinação de <i>n</i> e <i>m</i></p>
FIXED DECIMAL(15) Suportado no nível de mapeamento 3.0 e superior quando DATETIME=PACKED15	<pre>"type": "string", "format": "date-time"</pre> <p>O formato do registro de data e hora é definido pelo RFC3339.</p>
BIT(<i>n</i>) em que <i>n</i> é um múltiplo de 8. Outros valores não são suportados.	<pre>"type": "string" "maxLength": m</pre> <p>em que $m = n / 8$</p>
CHARACTER(<i>n</i>) VARYING e VARYINGZ também são suportados no nível de mapeamento 1.2 e superior. Restrição: VARYINGZ é suportado somente pelo PL/I Corporativo	<pre>"type": "string" "maxLength": n</pre>
GRAPHIC(<i>n</i>) VARYING e VARYINGZ também são suportados no nível de mapeamento 1.2 e superior. Restrição: VARYINGZ é suportado somente pelo Enterprise PL/I	<p>Em um nível de mapeamento 1.0 e 1.1, em que $m = 2 * n$:</p> <pre>"type": "string" "maxLength": m</pre> <p>Em um nível de mapeamento de 1.2 ou superior:</p> <pre>"type": "string" "maxLength": n</pre>

Descrição de Dados PL/I	Definição de esquema JSON
<p>WIDECHAR(<i>n</i>)</p> <p>Restrição: Somente PL/I corporativo</p>	<p>Em um nível de mapeamento 1.0 e 1.1, em que $m = 2 * n$:</p> <pre>"type": "string" "maxLength": m</pre> <p>Em um nível de mapeamento de 1.2 ou superior:</p> <pre>"type": "string" "maxLength": n</pre> <p>No nível de mapeamento 4.0 e superior, o CICS preenche o campo de estrutura de dados do aplicativo com dados 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>Restrição: Apenas PL/I corporativo</p>	<pre>"type": "integer", "minimum": 0, "maximum": 255</pre>
<p>BINARY FLOAT(<i>n</i>)</p> <p>em que $n \leq 21$</p> <p>Suportado no nível de mapeamento 1.2 e superior.</p> <p>Nota: A representação de dados do IBM Hexadecimal Floating Point (HFP) não é exatamente igual à representação IEEE-754-1985 usada para JSON. Alguns valores podem não ser convertidos exatamente de uma representação para a outra. Alguns valores extremamente grandes ou pequenos podem não ser válidos para tipos de dados flutuantes. Alguns valores podem perder a precisão quando convertidos para ou a partir de uma representação HFP. Se conversões precisas forem importantes, considere substituir o uso de tipos de dados BINARY FLOAT por alternativas de precisão fixas.</p>	<pre>"type": "number", "format": "float"</pre>

Descrição de Dados PL/I	Definição de esquema JSON
<p>BINARY FLOAT(<i>n</i>)</p> <p>em que $21 < n \leq 53$</p> <p>Valores maiores que 53 não são suportados.</p> <p>Suportado no nível de mapeamento 1.2 e superior.</p> <p>Nota: A representação de dados do IBM Hexadecimal Floating Point (HFP) não é exatamente igual à representação IEEE-754-1985 usada para JSON. Alguns valores podem não ser convertidos exatamente de uma representação para a outra. Alguns valores extremamente grandes ou pequenos podem não ser válidos para tipos de dados flutuantes. Alguns valores podem perder a precisão quando convertidos para ou a partir de uma representação HFP. Se conversões precisas forem importantes, considere substituir o uso de tipos de dados BINARY FLOAT por alternativas de precisão fixas.</p>	<pre>"type": "number", "format": "double"</pre>
<p>DECIMAL FLOAT(<i>n</i>)</p> <p>em que $n \leq 6$</p> <p>Suportado no nível de mapeamento 1.2 e superior.</p> <p>Nota: A representação de dados do IBM Hexadecimal Floating Point (HFP) não é exatamente igual à representação IEEE-754-1985 usada para JSON. Alguns valores podem não ser convertidos exatamente de uma representação para a outra. Alguns valores extremamente grandes ou pequenos podem não ser válidos para tipos de dados flutuantes. Alguns valores podem perder a precisão quando convertidos para ou a partir de uma representação HFP. Se conversões precisas forem importantes, considere substituir o uso de tipos de dados DECIMAL FLOAT por alternativas de precisão fixas.</p>	<pre>"type": "number", "format": "float"</pre>

Descrição de Dados PL/I	Definição de esquema JSON
<p>DECIMAL FLOAT(<i>n</i>)</p> <p>em que $6 < n \leq 16$</p> <p>Valores maiores que 16 não são suportados.</p> <p>Suportado no nível de mapeamento 1.2 e superior.</p> <p>Nota: A representação de dados do IBM Hexadecimal Floating Point (HFP) não é exatamente igual à representação IEEE-754-1985 usada para JSON. Alguns valores podem não ser convertidos exatamente de uma representação para a outra. Alguns valores extremamente grandes ou pequenos podem não ser válidos para tipos de dados flutuantes. Alguns valores podem perder a precisão quando convertidos para ou a partir de uma representação HFP. Se conversões precisas forem importantes, considere substituir o uso de tipos de dados DECIMAL FLOAT por alternativas de precisão fixas.</p>	<pre>"type": "number", "format": "double"</pre>
<p><i>name (n) data description</i></p>	<p>Para primitivas:</p> <pre>"type": "array" "maxItems": n "minItems": n "items": { "tipo": "objeto", "properties": { name : { data description JSON } } "required": [name] }</pre> <p>Para declarações de dados:</p> <pre>"type": "array" "maxItems": n "minItems": n "items": { data description JSON }</pre> <p>Em que <i>data description JSON</i> é a representação de esquema JSON da descrição de dados PL/I.</p>

Esquema JSON para mapeamento de PL/I:

O programa utilitário DFHJS2LS suporta mapeamentos entre esquemas JSON e estruturas de dados PL/I. Como o compilador PL/I Corporativo e compiladores PL/I mais antigos diferem, duas opções de linguagem são suportadas: PLI-ENTERPRISE e PLI-OTHER.

Regras para mapear nomes de elemento de esquema para PL/I

Os assistentes do CICS geram nomes exclusivos e válidos para variáveis PL/I a partir dos nomes de elemento de esquema usando as regras a seguir:

1. Caracteres diferentes de A-Z, a-z, 0-9, @, #, _ ou \$ são substituídos por ' X' inicial.

Por exemplo, monthly-total se torna monthlyXtotal.

É possível usar o parâmetro **MAPPING-OVERRIDES** para mudar o modo como outros caracteres são manipulados. Por exemplo, se você configurar o valor **HYPHENS-AS-UNDERSCORES**, qualquer hífen no esquema JSON será convertido em um sublinhado em vez de um X. Por exemplo, monthly-total se torna monthly_total.

2. Nomes duplicados no mesmo escopo se tornam exclusivos pela adição de um ou dois dígitos numéricos na segunda instância e nas instâncias subsequentes do nome.

Por exemplo, três instâncias de year se tornam year , year1 e year2.

3. Um esquema JSON especifica que uma variável possui variação de cardinalidade se possui um valor de "type" igual a "array" e as palavras-chave "minItems" e "maxItems" foram omitidas ou possuem valores diferentes. Se o esquema especifica que a variável possui variação de cardinalidade, os nomes de campo são criados com sufixos "_cont" e "_num".

Para obter mais informações, consulte “Matrizes variáveis de elementos no DFHJS2LS” na página 282.

4. Um esquema JSON especifica que uma variável é opcional se ela não aparece na matriz de palavra-chave "required" que está associada ao tipo de "object" de esquema JSON de inclusão. Para campos opcionais, um campo adicional é gerado com um sufixo _num incluído no nome de elemento. No tempo de execução, ele é zero para indicar que o valor estava ausente nos dados JSON e não zero se o valor estava presente nos dados JSON.

5. Os nomes de campo são limitados a 31 caracteres. Se um nome gerado, incluindo qualquer prefixo e sufixo, exceder esse comprimento, o nome de elemento será truncado.

O comprimento total do nome resultante é 31 caracteres ou menos.

Regras para mapear tipos de esquema para PL/I

DFHJS2LS mapeia valores de tipo de esquema para tipos de dados PL/I de acordo com a tabela a seguir. Além disso, observe os seguintes pontos:

- Se o parâmetro **MAPPING-LEVEL** for configurado como 1.2 ou superior e o parâmetro **CHAR-VARYING** for configurado como NULL, os dados de caractere de comprimento variável serão mapeados para sequências com terminação nula e um caractere extra será alocado para o terminador NULL.
- Se o parâmetro **MAPPING-LEVEL** for configurado como 1.2 ou superior e o parâmetro **CHAR-VARYING** não for especificado, por padrão, os dados de caractere de comprimento variável serão mapeados para um tipo de dados VARYINGZ para Enterprise PL/I e para o tipo de dados VARYING para Outro PL/I.

- Dados binários de comprimento variável são mapeados para um tipo de dados VARYING se são menores que 32.768 bytes e para um contêiner se são maiores que 32.768 bytes.

Palavra-chave do Esquema JSON	Descrição de Dados PL/I
Todos de: "type": "null" "type": [] "enum": [] "allOf" "anyOf" "noneOf" "not"	Não-suportado
"\$schema": "http://json-schema.org/draft-04/schema#"	Esta palavra-chave é ignorada, mas assume-se que ela é compatível com a especificação de Esquema JSON 04 de rascunho.
"title": "same text" "description": "more text"	Essas palavras-chave são ignoradas.
"format": "<predefined values>"	A palavra-chave "format" é usada para modificar a estrutura gerada ou o valor de tempo de execução. Consulte as informações a seguir para o uso suportado de "format".
"type": "array", "items": {<JSON Sub-schema>}, "additionalItems": false, "maxItems": <i>m</i> , "minItems": <i>n</i>	<p>A única forma de matriz JSON suportada atualmente é um número repetido dos mesmos valores de tipo. O <JSON Sub-schema> deve definir um "type" suportado , mas esse "type" não pode ser "array". Esta é uma restrição na estrutura de linguagem gerada.</p> <p>"additionalItems" é assumido como false e nenhum outro valor é suportado.</p> <p>Se "minItems" e "maxItems" estiverem presentes e forem iguais, a matriz será tratada como cardinalidade fixa, caso contrário, será tratada como cardinalidade variável. Consulte "Matrizes variáveis de elementos no DFHJS2LS" na página 282.</p>
"type": "array", "uniqueItems": true	"uniqueItems" não é suportado com matrizes JSON. O <JSON Sub-schema> deve definir um "type" suportado , mas esse "type" não pode ser "array" . Esta é uma restrição na estrutura de linguagem gerada.

Palavra-chave do Esquema JSON	Descrição de Dados PL/I
<pre>"type": "object", "additionalProperties": false, "properties": { ["<element name>": {<JSON Sub-schema> } [,]]* } "required": [["<element name>" [,]]*]</pre>	<p>A única forma de objeto JSON atualmente suportada é um conjunto fixo de elementos nomeados.</p> <p>Isso gerará uma estrutura (ou subestrutura) usando os nomes de elementos.</p> <p>"additionalProperties" é assumido como false e nenhum outro valor é suportado.</p> <p>Qualquer elemento no objeto "properties" é considerado "optional" se ele não está na matriz "required" ou se nenhuma matriz "required" existe. Um elemento "optional" recebe uma cardinalidade variável de zero a X; em que X é 1 ou o número máximo de itens na matriz no qual esse item é definido como uma matriz. Consulte "Matrizes variáveis de elementos no DFHJS2LS" na página 282.</p>
<pre>"type": "object", "maxProperties": m, "minProperties": n, "patternProperties": {}, "dependencies":</pre>	<p>Nenhuma dessas palavras-chave é suportada com objetos JSON.</p>
<pre>"type": "string" "maxLength": m "pattern": "regular expression", "minLength": l</pre>	<p>char[z]</p> <p>em que o valor de z é baseado em m, porém é dependente das configurações do parâmetro CHAR-VARYING.</p> <p>m é baseado na palavra-chave "maxLength" e tratado como uma sequência de comprimento fixo.</p> <p>As restrições "pattern" e "minLength" são transmitidas para a estrutura de linguagem somente como um comentário.</p>
<pre>"type": "string" "maxLength": m</pre>	<p>Quando CCSID=1200 no nível de mapeamento 4.0 e superior:</p> <p>WIDECHAR(z)</p> <p>em que o valor de z é baseado em m, porém é dependente das configurações do parâmetro CHAR-VARYING.</p> <p>m é baseado na palavra-chave "maxLength" e tratado como uma sequência de comprimento fixo.</p>

Palavra-chave do Esquema JSON	Descrição de Dados PL/I
<pre>"*name*":{ "type": "string", "format":"date-time" }</pre>	<p>FIXED DECIMAL (15,0)</p> <p>Todos suportados quando DATETIME=PACKED15</p>
<pre>"*name*":{ "type": "string", "format":"uri" }</pre>	<p>CHAR(<i>m</i>)</p> <p>em que <i>m</i> é baseado na palavra-chave "maxLength" e tratado como sequência de comprimento fixo.</p> <p>Quando CCSID=1200 no nível de mapeamento 4.0 e superior:</p> <p>WIDECHAR(<i>m</i>)</p> <p>em que <i>m</i> é baseado na palavra-chave "maxLength" e tratado como uma sequência de comprimento fixo.</p>
<pre>"*name*":{ "type": "string", "format":"base64Binary" }</pre>	<p>CHAR(<i>m</i>)</p> <p>em que <i>m</i> é baseado na palavra-chave "maxLength".</p>
<pre>"*name*":{ "type": "string", "format":"hexBinary" }</pre>	<p>CHAR(<i>m</i>)</p> <p>em que <i>m</i> é baseado na palavra-chave "maxLength".</p>
<pre>"*name*":{ "type": "string", "format":"<predefined>" }</pre>	<p>CHAR(<i>m</i>) em que <i>m</i> é baseado na palavra-chave "maxLength" e tratado como uma sequência de comprimento fixo e <predefined> é um de: <i>email</i>, <i>hostname</i>, <i>ipv4</i> ou <i>ipv6</i>. Um "pattern" relevante é transmitido ao comentário.</p> <p>Quando CCSID=1200 no nível de mapeamento 4.0 e superior:</p> <p>WIDECHAR(<i>m</i>)</p> <p>em que <i>m</i> é baseado na palavra-chave "maxLength" e tratado como sequência de comprimento fixo e em que <predefined> é um de: <i>e-mail</i>, <i>hostname</i>, <i>ipv4</i> ou <i>ipv6</i>. Um "padrão" relevante é usado e transmitido ao comentário.</p>

Palavra-chave do Esquema JSON	Descrição de Dados PL/I
"type": "boolean"	<p>Nível de mapeamento 1.1 e inferior:</p> <p>Enterprise PL/I SIGNED FIXED BINARY (7)</p> <p>Outro PL/I FIXED BINARY (7)</p> <p>Nível de mapeamento 1.2 e posterior:</p> <p>Enterprise PL/I BIT(7) BIT(1)</p> <p>Outro PL/I BIT(7) BIT(1)</p> <p>em que BIT(7) é fornecido para alinhamento e BIT(1) Contém o valor mapeado Booleano.</p>
"type": "integer", "maxExclusive": true, "minExclusive": true, "multipleOf": n	<p>As restrições "maxExclusive" e "minExclusive" são transmitidas à estrutura de linguagem somente como um comentário.</p> <p>"multipleOf" é ignorado.</p>
"type": "integer", minimum:-128, maximum:127	<p>Enterprise PL/I SIGNED FIXED BINARY (7)</p> <p>Outro PL/I FIXED BINARY (7)</p>
"type": "integer", minimum:0, maximum:255	<p>Enterprise PL/I UNSIGNED FIXED BINARY (8)</p> <p>Outro PL/I FIXED BINARY (8)</p>
"type": "integer", minimum:-32768, maximum:32767	<p>Enterprise PL/I SIGNED FIXED BINARY (15)</p> <p>Outro PL/I FIXED BINARY (15)</p>
"type": "integer", minimum:0, maximum:65535	<p>Enterprise PL/I UNSIGNED FIXED BINARY (16)</p> <p>Outro PL/I FIXED BINARY (16)</p>
"type": "integer", minimum:-2147483648, maximum:2147483647	<p>Enterprise PL/I SIGNED FIXED BINARY (31)</p> <p>Outro PL/I FIXED BINARY (31)</p>

Palavra-chave do Esquema JSON	Descrição de Dados PL/I
"type": "integer", minimum:0, maximum:4294967295	Nível de mapeamento 1.1 e inferior: Enterprise PL/I UNSIGNED FIXED BINARY(32) Nível de mapeamento 1.2 e posterior: Enterprise PL/I CHAR(<i>y</i>) em que <i>y</i> é um comprimento fixo menor do que 16 MB. Todos os níveis de mapeamento: Outro PL/I BIT(64)
"type": "integer", minimum:-9223372036854775808, maximum:9223372036854775807	Nível de mapeamento 1.1 e inferior: Enterprise PL/I SIGNED FIXED BINARY(63) Nível de mapeamento 1.2 e posterior: Enterprise PL/I CHAR(<i>y</i>) em que <i>y</i> é um comprimento fixo menor do que 16 MB. Todos os níveis de mapeamento: Outro PL/I BIT(64)
"type": "integer", minimum:0, maximum:18446744073709551615	Nível de mapeamento 1.1 e inferior: Enterprise PL/I UNSIGNED FIXED BINARY(64) Nível de mapeamento 1.2 e posterior: Enterprise PL/I CHAR(<i>y</i>) em que <i>y</i> é um comprimento fixo menor do que 16 MB. Outro PL/I BIT(64)
"type": "number" "description": "decimal"	FIXED DECIMAL(<i>n</i> , <i>m</i>)

Palavra-chave do Esquema JSON	Descrição de Dados PL/I
<pre>"type": "number" "description": "float"</pre>	<p>Níveis de Mapeamento 1.0 e 1.1:</p> <ul style="list-style-type: none"> CHAR(32) <p>Nível de mapeamento 1.2 e posterior:</p> <p>Enterprise PL/I DECIMAL FLOAT(6) HEXADEC</p> <p>Outro PL/I DECIMAL FLOAT(6)</p> <p>Nota: A representação de dados do IBM Hexadecimal Floating Point (HFP) não é exatamente igual à representação IEEE-754-1985 usada para JSON. Alguns valores podem não ser convertidos exatamente de uma representação para a outra. Alguns valores extremamente grandes ou pequenos podem não ser válidos para tipos de dados flutuantes. Alguns valores podem perder a precisão quando convertidos para ou a partir de uma representação HFP. Se conversões precisas forem importantes, considere substituir o uso de tipos de dados DECIMAL FLOAT por alternativas de precisão fixas.</p>
<pre>"type": "number" "description": "double"</pre>	<p>Níveis de Mapeamento 1.0 e 1.1:</p> <ul style="list-style-type: none"> CHAR(32) <p>Nível de mapeamento 1.2 e posterior:</p> <p>Enterprise PL/I DECIMAL FLOAT(16) HEXADEC</p> <p>Outro PL/I DECIMAL FLOAT(16)</p> <p>Nota: A representação de dados do IBM Hexadecimal Floating Point (HFP) não é exatamente igual à representação IEEE-754-1985 usada para JSON. Alguns valores podem não ser convertidos exatamente de uma representação para a outra. Alguns valores extremamente grandes ou pequenos podem não ser válidos para tipos de dados flutuantes. Alguns valores podem perder a precisão quando convertidos para ou a partir de uma representação HFP. Se conversões precisas forem importantes, considere substituir o uso de tipos de dados DECIMAL FLOAT por alternativas de precisão fixas.</p>

Nota: O CICS não pode transformar valores de número inteiro maiores que o valor máximo para um longo sinalizado ($2^{63} - 1$) a menos que eles sejam colocados entre aspas.

Nota: Os valores mínimo e máximo especificados no esquema para tipos numéricos são usados somente para mapear para um tipo de dados PL/I. Os dados não são validados com relação a esses valores no tempo de execução.

Matrizes de Elementos Variáveis

O XML pode conter uma matriz com números variados de elementos. Em geral, documentos WSDL e esquemas XML que contêm números variados de elementos não mapeiam eficientemente para uma única estrutura de dados de linguagem de alto nível. O CICS usa mapeamentos baseados em contêiner ou mapeamentos sequenciais para manipular números variados de elementos em XML.

Uma matriz com um número variado de elementos é representada no esquema XML usando os atributos `minOccurs` e `maxOccurs` na declaração de elemento:

- O atributo `minOccurs` especifica o número mínimo de vezes que o elemento pode ocorrer. Ela pode ter um valor 0 ou qualquer número inteiro positivo.
- O atributo `maxOccurs` especifica o número máximo de vezes que o elemento pode ocorrer. Ele pode ter um valor de qualquer número inteiro positivo maior ou igual ao valor do atributo `minOccurs`. Ele também pode ter um valor igual a `unbounded`, que indica que nenhum limite superior se aplica ao número de vezes que o elemento pode ocorrer.
- O valor padrão para ambos os atributos é 1.

Este exemplo denota uma sequência de 8 bytes que é opcional; ou seja, ela pode não ocorrer nunca ou ocorrer uma vez no XML do aplicativo ou na mensagem SOAP:

```
<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>
```

O exemplo a seguir denota uma sequência de 8 bytes que deve ocorrer pelo menos uma 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>
```

Em geral, documentos WSDL que contêm números variados de elementos não mapeiam eficientemente para uma única estrutura de dados de linguagem de alto nível. Portanto, para manipular esses casos, o CICS usa uma série de estruturas de dados conectadas que são transmitidas ao programa de aplicativo em uma série de contêineres. Essas estruturas são usadas como entrada e saída do aplicativo:

- Quando o CICS transforma XML em dados do aplicativo, ele preenche essas estruturas com os dados do aplicativo e o aplicativo os lê.
- Quando o CICS transforma os dados do aplicativo em XML, ele lê os dados do aplicativo nas estruturas que foram preenchidas pelo aplicativo.

O formato dessas estruturas de dados é explicado melhor com uma série de exemplos. O XML pode ser de uma mensagem SOAP ou de um aplicativo. Estes

exemplos usam uma matriz de campos de 8 bytes simples. No entanto, o modelo suporta matrizes de tipos de dados complexos e matrizes de tipos de dados que contêm outras matrizes.

Número Fixo de Elementos

O primeiro exemplo ilustra um elemento que ocorre exatamente três vezes:

```
<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>
```

Neste exemplo, como o número de vezes que o elemento ocorre é conhecido antecipadamente, ele pode ser representado como uma matriz de comprimento fixo em uma declaração COBOL simples (ou o equivalente em outras linguagens):

```
05 component PIC X(8) OCCURS 3 TIMES
```

Número variado de elementos no nível de mapeamento 2 e abaixo

Este exemplo ilustra um elemento obrigatório que pode ocorrer de uma a cinco vezes:

```
<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>
```

A estrutura de dados principal contém uma declaração de dois campos. Quando o CICS transforma o XML em dados binários, o primeiro campo component-num contém o número de vezes que o elemento aparece no XML e o segundo campo, component-cont, contém o nome de um contêiner:

```
05 component-num PIC S9(9) COMP-5
05 component-cont PIC X(16)
```

Uma segunda estrutura de dados contém a declaração do elemento em si:

```
01 DFHWS-component
02 component PIC X(8)
```

Você deve examinar o valor de component-num (que conterá um valor no intervalo de 1 a 5) para descobrir quantas vezes o elemento ocorre. O conteúdo do elemento está no contêiner nomeado em component-cont; o contêiner contém uma matriz de elementos, em que cada elemento é mapeado pela estrutura de dados DFHWS-component.

Se minOccurs="0" e maxOccurs="1", o elemento é opcional. Para processar a estrutura de dados em seu programa de aplicativo, você deve examinar o valor de component-num:

- Se for zero, a mensagem não possui elemento do componente e o conteúdo de component-cont é indefinido.

- Se for um, o elemento do componente está no contêiner nomeado em `component-cont`.

O conteúdo do contêiner é mapeado pela estrutura de dados DFHWS-component.

Nota: Se a mensagem SOAP consistir em um único elemento recorrente, o DFHWS2LS gerará duas estruturas de linguagem. A estrutura de linguagem principal contém o número de elementos na matriz e o nome de um contêiner que contém a matriz de elementos. A segunda estrutura de linguagem mapeia uma única instância do elemento recorrente.

Número variado de elementos no nível de mapeamento 2.1 e superior

No nível de mapeamento 2.1 e acima, é possível usar o parâmetro **INLINE-MAXOCCURS-LIMIT** nos assistentes do CICS. O parâmetro **INLINE-MAXOCCURS-LIMIT** especifica a maneira como números variados de elementos são manipulados. As opções de mapeamento para números variados de elementos são mapeamento baseado em contêiner, descrito em “Número variado de elementos no nível de mapeamento 2 e abaixo” na página 348, ou mapeamento sequencial. O *value* deste parâmetro pode ser um número inteiro positivo no intervalo de 0 a 32767:

- O valor padrão de **INLINE-MAXOCCURS-LIMIT** é 1, o qual assegura que elementos opcionais sejam mapeados sequencialmente.
- Um valor 0 para o parâmetro **INLINE-MAXOCCURS-LIMIT** evita o mapeamento sequencial.
- Se `maxOccurs` for menor ou igual ao valor de **INLINE-MAXOCCURS-LIMIT**, o mapeamento sequencial será usado.
- Se `maxOccurs` for maior do que o valor de **INLINE-MAXOCCURS-LIMIT**, o mapeamento baseado em contêiner será usado.

O mapeamento de números variados de elementos sequenciais resulta na geração de uma matriz, como acontece com o exemplo de ocorrência fixa acima, e de um contador. O campo `component-num` indica quantas instâncias do elemento estão presentes e elas são apontadas pela matriz. Para o exemplo mostrado em “Número variado de elementos no nível de mapeamento 2 e abaixo” na página 348, quando **INLINE-MAXOCCURS-LIMIT** é menor ou igual a 5, a estrutura de dados gerada conforme a seguir:

```
05 component-num PIC S9(9) COMP-5 SYNC.
05 component OCCURS 5 PIC X(8).
```

O primeiro campo, `component-num`, é idêntico ao de saída para o mapeamento baseado em contêiner de exemplo na seção anterior. O segundo campo contém uma matriz de comprimento 5 que é grande o suficiente para conter o número máximo de elementos que podem ser gerados.

O mapeamento sequencial difere do mapeamento baseado em contêiner, que armazena o número de ocorrências do elemento e o nome do contêiner no qual os dados são colocados, porque ele armazena todos os dados no contêiner atual. O armazenamento dos dados no contêiner atual geralmente irá melhorar o desempenho e tornar o mapeamento sequencial preferível.

Matrizes de Variáveis Aninhadas

Os documentos WSDL e esquemas XML complexos podem conter elementos variavelmente recorrentes, que por sua vez contêm os elementos variavelmente recorrentes. Nesse caso, a estrutura descrita se estende além dos dois níveis descritos nos exemplos.

Este exemplo ilustra um elemento opcional chamado <component2> que está aninhado em um elemento obrigatório chamado <component1>, em que o elemento obrigatório pode ocorrer de uma a cinco vezes:

```
<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>
```

A estrutura de dados de nível superior é exatamente a mesma que nos exemplos anteriores:

```
05 component1-num PIC S9(9) COMP-5
05 component1-cont PIC X(16)
```

No entanto, a segunda estrutura de dados contém estes elementos:

```
01 DFHWS-component1
02 component2-num PIC S9(9) COMP-5
02 component2-cont PIC X(16)
```

Uma estrutura de terceiro nível contém estes elementos:

```
01 DFHWS-component2
02 component2 PIC X(8)
```

O número de ocorrências do elemento <component1> mais externo está em component1-num.

O contêiner nomeado em component1-cont contém uma matriz com esse número de instâncias da segunda estrutura de dados DFHWS-component1.

Cada instância de component2-cont nomeia um contêiner diferente, cada um dos quais contém a estrutura de dados mapeada pela estrutura de terceiro nível DFHWS-component2.

Para ilustrar essa estrutura, considere o fragmento de XML que corresponde ao exemplo:

```
<component1><component2>
string1
</component2></component1>
<component1><component2>
string2
</component2></component1>
<component1></component1>
```


<component1> ocorre três vezes. Cada um dos dois primeiros contém uma instância de <component2>; a terceira instância não.

Na estrutura de dados de nível superior, component1-num contém um valor igual a 3. O contêiner nomeado em component1-cont possui três instâncias de DFHWS-component1 :

1. Na primeira, component2-num possui um valor 1 e o contêiner nomeado em component2-cont contém *string1*.
2. Na segunda, component2-num possui um valor 1 e o contêiner nomeado em component2-cont contém *string2*.
3. Na terceira, component2-num possui um valor 0 e o conteúdo de component2-cont é indefinido.

Neste exemplo, a estrutura de dados completa é representada por quatro contêineres:

- A estrutura de dados raiz no contêiner DFHWS-DATA
- O contêiner nomeado em component1-cont
- Dois contêineres nomeados nas duas primeiras instâncias de component2-cont

Estruturas Opcionais e xsd:choice

DFHWS2LS e DFHSC2LS suportam o uso de maxOccurs e minOccurs nos elementos <xsd:sequence>, <xsd:choice> e <xsd:all> somente no nível de mapeamento 2.1 e acima, em que os atributos minOccurs e maxOccurs são configurados como minOccurs="0" e maxOccurs="1".

Os assistentes geram mapeamentos que tratam estes elementos como se cada elemento filho neles fosse opcional. Ao implementar um aplicativo com estes elementos, assegure que combinações inválidas de opções não sejam geradas pelo aplicativo. Cada um dos elementos possui seu próprio campo count na estrutura de linguagens gerada, todos estes campos devem ser configurados como "0" ou todos devem ser configurados como "1". Qualquer outra combinação de valores é inválida, exceto para elementos <xsd:choice>.

Elementos <xsd:choice> indicam que somente uma das opções no elemento pode ser usada. Isto é suportado em todos os níveis de mapeamento. Os assistentes manipulam cada uma das opções em um <xsd:choice> como se estivesse em um elemento <xsd:sequence> com minOccurs="0" e maxOccurs="1". Tome cuidado ao implementar um aplicativo usando o elemento <xsd:choice> para assegurar que combinações inválidas de opções não sejam geradas pelo aplicativo. Cada um dos elementos possui seu próprio campo count na estrutura de linguagens gerada, exatamente um dos quais deve ser configurado como '1' e todos os outros devem ser configurados como '0'. Qualquer outra combinação de valores é inválida, exceto quando o elemento <xsd:choice> é em si opcional, nesse caso ele é válido para todos os campos a serem configurados como '0'.

Suporte para Valores de Comprimento Variável e Espaço em Branco

É possível customizar a maneira na qual os valores de comprimento variável e espaços em branco são manipulados usando as configurações nos assistentes CICS e incluindo máscaras diretamente no esquema XML.

Geralmente, o assistente CICS XML e o assistente de serviço da web do CICS mapeiam sequências de dados para matrizes de caracteres de comprimento fixo; essas matrizes requerem preenchimento com espaços ou nulos. O mapeamento de

valores de comprimento variável para matrizes de dados de comprimento fixo pode ser ineficiente e desperdiçar armazenamento. Se o comprimento de seus dados é variável, é recomendável customizar a maneira como esses mapeamentos são manipulados.

Se você estiver convertendo a partir de uma estrutura de linguagem para um esquema XML ou documento WSDL, é recomendável especificar as máscaras `whiteSpace` e `maxLength` em seu esquema XML e configurar o parâmetro **CHAR-VARYING-LIMIT** nos assistentes.

Se você estiver convertendo a partir de um esquema XML ou documento WSDL para uma estrutura de linguagem, é recomendável configurar um valor apropriado para o parâmetro **CHAR-VARYING** nos assistentes.

Nota: Caracteres nulos ('x00') não são válidos em documentos XML. Quaisquer caracteres nulos de dados do aplicativo analisados pelo CICS são vistos como o fim de uma sequência e o valor é truncado. Quando o CICS gera dados do aplicativo, ele faz isso de acordo com o valor do parâmetro **CHAR-VARYING**. Por exemplo, se a opção **CHAR-VARYING=NULL** for especificada, as sequências de comprimento variável geradas pelo CICS serão finalizadas com um caractere nulo.

Mapeando valores de comprimento variável a partir do XML para estruturas de linguagem

Use máscaras no esquema XML ou especifique determinados parâmetros nos assistentes CICS para customizar a maneira na qual os mapeamentos entre o esquema XML ou o documento WSDL e a estrutura de linguagem são manipulados.

Tipos de dados XML podem ser restringidos usando máscaras. Use as máscaras de comprimento (`length`, `maxLength` e `minLength`) e a máscara `whiteSpace` para customizar a maneira como os dados de comprimento variável em seu XML são manipulados.

length

Usado para especificar que os dados são de comprimento fixo.

maxLength

Usado para especificar o comprimento máximo para o tipo de dados. Se esse valor não for configurado para um tipo de dados baseado em sequência, o comprimento máximo será ilimitado.

minLength

Usado para especificar o comprimento mínimo para o tipo de dados. Se esse valor não for configurado para um tipo de dados baseado em sequência, o comprimento mínimo será 0.

whiteSpace

Usado para especificar como o espaço em branco ao redor de um valor de dados é manipulado. O espaço em branco inclui espaços, tabulações e novas linhas. A máscara `whiteSpace` pode ser configurada como `preserve`, `replace` ou `collapse`:

- Um valor `preserve` mantém qualquer espaço em branco no valor dos dados.
- Um valor `replace` significa que quaisquer tabulações ou novas linhas serão substituídas pelo número apropriado de espaços.

- Um valor `collapse` significa que espaços em branco iniciais, finais e integrados são removidos e que todas as tabulações, novas linhas e espaços consecutivos são substituídos por caracteres de espaço único.

Se a máscara `whiteSpace` não for configurada, o espaço em branco será preservado.

Para obter mais informações sobre as máscaras de esquema XML, consulte o esquema de recomendação do W3C *Esquema XML Parte 2: Tipos de Dados Segunda Edição* <http://www.w3.org/TR/xmlschema-2/#facets>

Os parâmetros a seguir nos assistentes CICS, DFHSC2LS e DFHWS2LS, podem ser usados para alterar a maneira como os dados de comprimento variável são mapeados a partir do esquema XML para a estrutura de linguagem. Esses parâmetros estão disponíveis no nível de mapeamento 1.2 ou superior.

DEFAULT-CHAR-MAXLENGTH

Especifica o comprimento da matriz padrão de dados de caractere em caracteres para mapeamentos em que nenhum comprimento está implícito no esquema XML ou documento WSDL. O valor desse parâmetro pode ser um número inteiro positivo no intervalo de 1 a 2147483647.

No entanto, é recomendável especificar o comprimento máximo de caracteres que você deseja que DFHSC2LS ou DFHWS2LS use diretamente em seu esquema XML ou documento WSDL com a máscara `maxLength`. Especificar o comprimento máximo diretamente no esquema XML ou documento WSDL evita problemas associados a ter um padrão global aplicado a todos os tipos de dados baseados em sequência.

CHAR-VARYING-LIMIT

Especifica o tamanho máximo dos dados de caractere de comprimento variável que são mapeados para a estrutura de linguagem. Se os dados de caractere forem maiores que o valor especificado neste parâmetro, eles serão mapeados para um contêiner e o nome do contêiner será usado na estrutura de linguagem gerada. O valor pode variar de 0 até o padrão de 32767 bytes.

CHAR-VARYING

Especifica como os dados de caracteres de comprimento variável são mapeados. Se você não especificar esse parâmetro, o mapeamento padrão dependerá da linguagem especificada. Você pode selecionar estas opções:

- **CHAR-VARYING=NO** especifica que os dados de caracteres de comprimento variável são mapeados como sequências de comprimento fixo.
- **CHAR-VARYING=NULL** especifica que dados de caracteres de comprimento variável são mapeados para sequências com terminação nula.
- **CHAR-VARYING=YES** especifica que dados de caracteres de comprimento variável são mapeados para um tipo de dados CHAR VARYING em PL/I. Nas linguagens COBOL, C e C++, os dados de caracteres de comprimento variável são mapeados para uma representação equivalente que consiste em dois elementos relacionados: o comprimento dos dados e os dados.

A configuração de **CHAR-VARYING=YES** geralmente resulta no melhor desempenho.

Mapeando valores de comprimento variável a partir de estruturas de linguagem para XML

É possível customizar a maneira na qual os mapeamentos entre a estrutura de linguagem e o esquema XML, ou documento WSDL, são manipulados. Configure o parâmetro **CHAR-VARYING** em DFHLS2SC ou DFHLS2WS para COLLAPSE ou NULL para mudar a maneira como as matrizes de caracteres são geradas.

A configuração da opção **CHAR-VARYING=NULL** informa ao CICS para incluir um caractere nulo no final de cada matriz de caracteres ao gerar XML.

A configuração da opção **CHAR-VARYING=COLLAPSE** informa ao CICS para remover automaticamente quaisquer espaços à direita do final das matrizes de caracteres ao gerar o XML. Essa opção está disponível somente no nível de mapeamento 2.1 ou superior e **CHAR-VARYING=COLLAPSE** é o valor padrão no nível de mapeamento 2.1 ou superior para todas as linguagens diferentes de C e C++. Quando o XML é analisado, todos os espaços em branco iniciais, finais e integrados são removidos.

Para obter mais informações, consulte Suporte para espaço em branco e valores de comprimento variável nos serviços da web do CICS (nota técnica) .

Suporte para UTF-16 em dados do aplicativo

Os serviços da web CICS suportam a conversão de dados do aplicativo codificados em UTF-16 em XML ou JSON e também XML ou JSON em dados do aplicativo codificados em UTF-16. Use o UTF-16 quando você precisa armazenar e processar dados em diversos idiomas.

Os serviços da web CICS SOAP e JSON suportam a conversão de dados do aplicativo codificados em UTF-16 em XML ou JSON e também XML ou JSON em dados do aplicativo codificados em UTF-16. O Unicode é um esquema de codificação de largura variável que permite que os sistemas manipulem dados de forma eficiente.

UTF-16 é uma codificação de largura variável para Unicode, em que cada caractere é representado por 2 ou 4 bytes. Os serviços da web do CICS suportam CCSID 1200 para dados do aplicativo, que são UTF-16 BE (big endian) com a Área de Uso Privado da IBM. Esse comportamento é consistente com o suporte UTF-16 em todas as linguagens suportadas.

UTF-16 é suportado no nível de mapeamento 4.0 e superior. É possível customizar como os dados do aplicativo são convertidos usando configurações de mapeamento nos assistentes. Para obter mais informações sobre níveis de mapeamento XML, consulte Níveis de mapeamento para os assistentes CICS. Para obter mais informações sobre níveis de mapeamento JSON, consulte Níveis de mapeamento para os assistentes CICS JSON.

Nota: UTF-16 requer mais tempo de processamento e tem menos eficiência de armazenamento do que as codificações EBCDIC. Além disso, a combinação de tipos de codificação incorre em processamento de tempo de execução extra.

Mapeando UTF-16 do esquema XML ou JSON para estruturas de linguagem

O suporte para UTF-16 depende de como você cria o serviço da web. O mapeamento de esquema XML ou JSON para estruturas de linguagem, também conhecido como mapeamento de cima para baixo, tem as características a seguir. Se

UTF-16 está ativado, todos os campos de texto são mapeados para campos UTF-16, enquanto que tipos de dados de exibição numéricos em COBOL são mapeados como EBCDIC. Para usar UTF-16, configure o parâmetro CCSID de DFHJS2LS, DFHSC2LS ou DFHWS2LS para 1200.

Por exemplo, se o fragmento de esquema XML a seguir estiver presente no WSDL:

```
<xsd:element name="myString" nillable="false">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:maxLength value="20"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

O assistente DFHWS2LS gerará o campo a seguir em uma estrutura de linguagem COBOL:

```
myString PIC N(
20
) USAGE NATIONAL
```

O parâmetro CHAR-MULTIPLIER dos assistentes de serviços da web pode ser usado para especificar o comprimento de um campo que os assistentes geram.

CHAR-MULTIPLIER

Quando você usa UTF-16, os únicos valores válidos para o parâmetro **CHAR-MULTIPLIER** são 2 ou 4, em que 2 é o valor padrão.

CHAR-MULTIPLIER = 2, em que o esquema descreve uma sequência de $\text{maxlength} \times$, gera PIC N(x). Configurar **CHAR-MULTIPLIER** = 2 não impede o uso de pares substitutos em uma sequência UTF-16, mas afeta o número de caracteres que se ajustam no campo.

CHAR-MULTIPLIER = 4 gera PIC N($2x$). Se **CHAR-MULTIPLIER** = 4, o valor no tempo de execução é preenchido se a sequência inclui caracteres que podem ser expressos em uma única unidade de codificação.

Mapeando UTF-16 a partir de estruturas de linguagem para esquema XML ou JSON

O mapeamento de uma estrutura de linguagem para o esquema XML ou JSON, também conhecido como mapeamento bottom-up, é gerenciado de forma diferente do mapeamento de cima para baixo. Se uma sequência UTF-16 for declarada na estrutura de linguagem, os dados serão interpretados pelo CICS como codificados em UTF-16, caso contrário, assume-se que os dados estejam em uma codificação EBCDIC. O parâmetro CCSID para DFHLS2JS, DFHLS2SC ou DFHLS2WS indica a codificação de qualquer texto EBCDIC nos dados do aplicativo; ele não deve ser configurado para indicar UTF-16.

Os tipos de dados que são interpretadas como caracteres UTF-16 são os seguintes: PIC N(n) em COBOL, WIDECHAR(n) em PL/I e char16_t[n] em C e C++.

O parâmetro CHAR-USAGE dos assistentes de serviços da web pode ser usado para especificar os tipos de dados.

CHAR-USAGE

Em COBOL, o tipo de dado nacional, PIC N, pode ser usado para dados UTF-16 ou DBCS. Essa configuração é controlada pela opção do

compilador NSYMBOL. Deve-se configurar o parâmetro **CHAR-USAGE** no assistente com o mesmo valor que a opção do compilador NSYMBOL para assegurar que os dados sejam manipulados apropriadamente. Isso geralmente é configurado como CHAR-USAGE=NATIONAL quando você usa UTF-16.

Se você deseja combinar tipos de dados nacionais que contêm dados UTF-16 e DBCS no mesmo copybook, é possível usar os qualificadores USAGE NATIONAL ou USAGE DISPLAY-1 em campos individuais.

Nota: DFHLS2WS, DFHLS2SC e DFHLS2JS não suportam a cláusula COBOL GROUP USAGE NATIONAL.

Criando um Provedor de Serviço da Web Usando o Assistente de Serviços da Web

É possível criar um aplicativo do provedor de serviços a partir de uma descrição de serviços da web que está em conformidade com o WSDL 1.1 ou WSDL 2.0 ou a partir de uma estrutura de dados de linguagem de alto nível. O assistente de serviços da web do CICS ajuda a implementar seus aplicativos CICS em uma configuração do provedor de serviços.

Sobre Esta Tarefa

Quando você usa o assistente para implementar um aplicativo CICS como um provedor de serviços, existem duas opções:

- Iniciar com uma descrição de serviços da web e usar o assistente para gerar as estruturas de dados de linguagem.

Use esta opção quando você estiver implementando um provedor de serviços que esteja em conformidade com uma descrição de serviços da web existente.

- Iniciar com as estruturas de dados de linguagem e usar o assistente para gerar a descrição de serviços da web.

Use esta opção quando estiver expondo um programa existente como um serviço da web e estiver disposto a expor aspectos das interfaces de programa na descrição do serviço da web e nas mensagens SOAP.

É possível expor a descrição de serviços da web associada ao seu provedor de serviços usando um URI. Este URI tem o mesmo caminho que o URI associado ao WEBSERVICE com o sufixo ?wsdl anexado. Isso permite que os solicitantes dentro de seus negócios, ou externos a ele, descubram os arquivos WSDL associados aos seus provedores de serviços.

Criando um aplicativo do provedor de serviços a partir de uma descrição de serviços da web

Usando o assistente de serviços da web do CICS, é possível criar um aplicativo do provedor de serviços a partir de uma descrição de serviços da web que está em conformidade com WSDL 1.1 ou WSDL 2.0.

Antes de Iniciar

Antes de poder criar um aplicativo do provedor de serviços, as condições a seguir devem ser satisfeitas:

- Sua descrição de serviços da web deve estar em um arquivo UNIX no z/OS e você deve criar um pipeline do modo de provedor adequado na região do CICS.

- Você deve definir como OMVS o ID do usuário sob o qual DFHWS2LS é executado.
- O ID do usuário deve ter permissão de leitura para z/OS UNIX e bibliotecas PDS e permissão de gravação para os diretórios especificados nos parâmetros **LOGFILE**, **WSBIND** e **WSDL**.
- Você deve alocar armazenamento suficiente para o ID do usuário para que o ID execute Java. É possível usar qualquer versão suportada de Java. Por padrão, DFHWS2LS usa a versão de Java especificada no parâmetro **JAVADIR**.

Sobre Esta Tarefa

É possível usar o assistente de serviço da web para criar estruturas de linguagem a partir de seu WSDL para o aplicativo do provedor de serviços. Também é possível usar um documento WSDL que é armazenado em um servidor IBM WebSphere Service Registry and Repository (WSRR).

Procedimento

1. Use o programa em lote DFHWS2LS para gerar um arquivo de ligação de serviço da web e uma ou mais estruturas de dados de linguagem. O DFHWS2LS contém um grande conjunto de parâmetros opcionais que fornecem a flexibilidade para criar o arquivo de ligação e estruturas de linguagem que seu aplicativo requer. Considere essas opções quando ativar um aplicativo existente para serviços da web:
 - a. **Qual mecanismo o CICS usará para transmitir dados para o programa de aplicativo do provedor de serviços?**
 - É possível usar canais e transmitir os dados em contêineres ou usar uma COMMAREA. Os canais e contêineres são recomendados. Especifique-os com o parâmetro **PGMINT**.
 - b. **Qual linguagem deseja gerar?**
 - O DFHWS2LS pode gerar estruturas de dados de linguagem COBOL, C/C++ ou PL/I. Especifique a linguagem usando o parâmetro **LANG**.
 - c. **Qual nível de mapeamento deseja usar?**
 - Quanto maior o nível de mapeamento, mais controle e apoio você tem disponíveis para a manipulação de dados de caracteres e binários no tempo de execução. Alguns parâmetros opcionais estão disponíveis somente nos níveis de mapeamento mais altos. É recomendável usar o nível mais alto de mapeamento disponível. Especifique o nível de mapeamento com o parâmetro **MAPPING-LEVEL**.
 - d. **Qual URI deseja que o solicitante de serviço da web use?**
 - Especifique um URI relativo usando o parâmetro **URI**; por exemplo, `URI=/my/test/webservice`. O valor é usado pelo CICS quando ele cria o recurso URIMAP.
 - e. **Sob qual transação e ID do usuário você executará a solicitação e a resposta de serviço da web?**
 - É possível usar uma transação de alias para executar o aplicativo para compor uma resposta para o solicitante de serviço. A transação de alias é conectada sob o ID do usuário.
 - Especifique-a com os parâmetros **TRANSACTION** e **USERID**. Estes valores são usados ao criar o recurso URIMAP. Se você não desejar usar uma transação específica, não use esses parâmetros.
 - f. **Onde o documento WSDL é armazenado?**

- Se você deseja recuperar um documento WSDL a partir de um servidor WSRR, em vez de a partir do sistema de arquivos local, deverá especificar determinados parâmetros no DFHWS2LS.
 - No mínimo, deve-se especificar o parâmetro **WSRR-SERVER** com o local do servidor WSRR e o parâmetro **WSRR-NAME** com o nome do documento WSDL que você deseja recuperar a partir do WSRR.
 - Para obter informações sobre outros parâmetros que você possa desejar especificar se estiver usando o WSRR, consulte “DFHWS2LS: WSDL para Conversão de Linguagem de Alto Nível” na página 504.
- g. **Se você pretende recuperar seu documento WSDL a partir de um servidor WSRR, deseja fazer isso usando uma conexão segura?**
- É possível usar a criptografia secure socket layer (SSL) configurando os parâmetros apropriados para que interoperem com segurança com o WSRR. Para obter um exemplo, consulte “Exemplo de Como Usar SSL com o Assistente de Serviços da Web e o WSRR” na página 51.
 - Ao enviar DFHWS2LS, o CICS gera o arquivo de ligação de serviço da web e o coloca no local que você especificou com o parâmetro **WSBIND**. As estruturas de linguagem são colocadas no conjunto de dados particionados que você especificou com o parâmetro **PDSLIB**.
2. Copie o arquivo de ligação de serviço da web gerado no diretório pickup do recurso PIPELINE do modo de provedor que você deseja usar para seu aplicativo de serviço da web. Deve-se copiar o arquivo de ligação no modo binário.
 3. Opcional: Copie a descrição de serviços da web ou o archive que contém um ou mais descrições de serviços da web no mesmo diretório que o arquivo de ligação de serviço da web. O archive deve ser um arquivo .zip e o nome do arquivo deve corresponder ao nome do arquivo WSDL. Com esta cópia, é possível descobrir o WSDL.
 4. Grave um programa de aplicativo do provedor de serviços na interface com as estruturas de linguagem geradas e implemente a lógica de negócios necessária.
 5. Crie o recurso WEBSERVICE e dois recursos URIMAP.
 - O recurso WEBSERVICE encapsula o arquivo de ligação de serviço da web no CICS e é usado no tempo de execução.
 - O primeiro recurso URIMAP fornece ao CICS as informações para associar o recurso WEBSERVICE a um URI específico.
 - O segundo recurso URIMAP fornece ao CICS as informações para associar o archive WSDL ou documento WSDL a um URI específico.
 - Este URI tem o mesmo caminho que o URI associado ao WEBSERVICE com o sufixo ?wsdl anexado.
 - Este recurso URIMAP é criado para que os solicitantes externos possam usar o URI para descobrir o archive WSDL ou o documento WSDL.
 - Este recurso URIMAP é criado somente se a descrição de serviços da web ou o archive que contém uma ou mais descrições de serviços da web foi copiado no mesmo diretório que o arquivo de ligação de serviço da web.
 - Se o diretório de recebimento contiver um archive WSDL e um documento WSDL, o URI retornará somente o WSDL no archive.
 - Esta função está disponível somente para serviços da web instalados usando a operação de varredura de pipeline.
- É possível criar os recursos das maneiras a seguir:
- a. Usando o comando **PIPELINE SCAN** para criar dinamicamente o recurso WEBSERVICE e os recursos URIMAP.

- b. Definindo os recursos você mesmo. Se você usar o CICS Explorer para definir um recurso WEBSERVICE em um pacote configurável do CICS, poderá optar por importar um arquivo de ligação de serviço da web e um documento WSDL ou archive WSDL e incluí-los no pacote configurável. É possível, então, gerar as definições de URIMAP para suportar o serviço da web e empacotá-las em um pacote configurável. Para obter mais ajuda com o uso do CICS Explorer para criar e editar recursos nos pacotes configuráveis do CICS, consulte Trabalhando com pacotes configuráveis na documentação do produto CICS Explorer.

Resultados

Se você tiver algum problema ao enviar o DFHWS2LS, ou se os recursos não forem instalados corretamente, consulte Diagnosticando erros de implementação.

Criando um Aplicativo do Provedor de Serviços a Partir de uma Estrutura de Dados

Usando o assistente de serviços da web do CICS, é possível criar um aplicativo do provedor de serviços a partir de uma estrutura de dados de linguagem de alto nível.

Antes de Iniciar

Antes de criar um aplicativo do provedor de serviços, certifique-se de que estas condições prévias tenham sido concluídas:

- Suas estruturas de dados de linguagem de alto nível devem atender aos seguintes critérios:
 - As estruturas de dados devem ser definidas separadamente do programa de origem; por exemplo, em um copybook COBOL.
 - Se o seu programa de aplicativo PL/I ou COBOL usar estruturas de dados diferentes para entrada e saída, as estruturas de dados deverão ser definidas em dois membros diferentes em um conjunto de dados particionados. Se a mesma estrutura for usada para entrada e saída, a estrutura deverá ser definida em um único membro.
Para C e C++, suas estruturas de dados podem estar no mesmo membro em um conjunto de dados particionados.
- As estruturas de dados que você processa dependem de se você está usando um programa wrapper:
 - Se você estiver usando um programa wrapper, o copybook será a interface para o programa wrapper.
 - Se você não estiver usando um programa wrapper, o copybook será a interface para a lógica de negócios.
- As estruturas de linguagem devem estar disponíveis em um conjunto de dados particionados e você deve criar um recurso PIPELINE adequado na região CICS:
 - Você deve definir para OMVS o ID do usuário sob o qual DFHLS2WS é executado.
 - O ID do usuário deve ter permissão de leitura para z/OS UNIX e bibliotecas PDS e permissão de gravação para os diretórios especificados nos parâmetros **LOGFILE**, **WSBIND** e **WSDL**.

- O ID do usuário deve ter uma alocação de armazenamento suficientemente grande para executar Java. É possível usar qualquer versão suportada do Java. Por padrão, o DFHLS2WS usa a versão de Java especificada no parâmetro **JAVADIR**.

Procedimento

Siga estas etapas para criar um aplicativo provedor de serviços a partir de uma estrutura de dados:

1. Se a interface de aplicativo do provedor de serviços usar canais e muitos contêineres, crie um documento de descrição de canal que descreva a interface em XML. Deve-se colocar o documento de descrição de canal em um diretório adequado no z/OS UNIX. O CICS usa este documento para construir e desconstruir uma mensagem SOAP a partir dos contêineres em um canal. Como alternativa, você pode usar um contêiner em um canal e não criar um documento de descrição de canal.
 - Para obter mais informações sobre como criar um documento de descrição de canal, consulte “Criando um Documento de Descrição de Canal” na página 580.
2. Use o programa em lote DFHLS2WS para gerar um arquivo de ligação de serviço da web e a descrição de serviços da web a partir da estrutura de linguagem. O DFHLS2WS contém um grande conjunto de parâmetros opcionais que fornecem a flexibilidade para criar o arquivo de ligação e as estruturas de linguagem que seu aplicativo requer. Considere estas opções quando o serviço da web ativar um aplicativo existente:
 - a. **Qual mecanismo o CICS usará para transmitir dados para o programa de aplicativo do provedor de serviços?**
 - É possível usar canais e transmitir os dados em contêineres ou usar uma COMMAREA. Especifique o mecanismo usando o parâmetro **PGMINT**. Se sua interface de aplicativo usar canais e muitos contêineres, especifique o parâmetro **REQUEST-CHANNEL** e, opcionalmente, o **RESPONSE-CHANNEL**. Só é possível usar esses parâmetros quando o nível de mapeamento é 3.0 ou superior.
 - b. **Qual nível de descrição de serviços da web (documento WSDL) você deseja gerar?**
 - O CICS gera descrições que estão em conformidade com os documentos do WSDL 1.1 ou WSDL 2.0. Se você deseja que o aplicativo do provedor de serviços suporte solicitações que estão em conformidade com os níveis de WSDL, especifique valores para os parâmetros **WSDL_1.1** e **WSDL_2.0**. Assegure que os nomes de arquivos sejam diferentes ao usar mais de um parâmetro WSDL. Esta especificação produz duas descrições de serviços da web e um arquivo de ligação.
 - c. **Qual versão do protocolo SOAP você deseja usar?**
 - É possível especificar a versão com o parâmetro **SOAPVER**. É recomendável que você use o valor ALL, que fornece a flexibilidade para usar SOAP 1.1 ou SOAP 1.2 como a ligação para a descrição de serviços da web, embora seja necessário instalar o serviço da web em um pipeline que esteja configurado com o manipulador de mensagem SOAP 1.2. É possível usar esse parâmetro somente quando o **MINIMUM-RUNTIME-LEVEL** é 2.0 ou superior.
 - d. **Qual nível de mapeamento deseja usar?**
 - Quanto maior o nível de mapeamento, mais controle e apoio você tem disponíveis para a manipulação de dados de caracteres e binários no

tempo de execução. Alguns parâmetros opcionais estão disponíveis somente nos níveis de mapeamento mais altos. É recomendável especificar o nível mais alto de mapeamento disponível no parâmetro **MAPPING-LEVEL**.

- e. **Qual URI você deseja que o solicitante de serviço da web use?**
 - Especifique um URI absoluto usando o parâmetro **URI**; por exemplo, **URI** = `http://www.example.org:80/my/test/webservice`. A parte relativa deste endereço, `/my/test/webservice`, é usada ao criar o recurso URIMAP. O URI completo é usado como o elemento `<soap:address>` na descrição do serviço da web. Este uso é verdadeiro para URIs HTTP e do WebSphere MQ.
- f. **Deseja publicar seu documento WSDL em um IBM WebSphere Service Registry and Repository (WSRR)?**
 - Se você deseja publicar seu documento WSDL em um WSRR, deverá especificar o parâmetro **WSRR-SERVER** no DFHLS2WS. Para obter mais informações sobre os parâmetros que você pode especificar ao usar o WSRR, consulte “DFHLS2WS: Linguagem de Alto Nível para Conversão WSDL” na página 488.
- g. **Se você pretende publicar seu documento WSDL em um servidor WSRR, deseja fazer isso usando uma conexão segura?**
 - É possível usar a criptografia secure socket layer (SSL) configurando os parâmetros apropriados para que interoperem com segurança com o WSRR. Para obter um exemplo, consulte Exemplo de Como Usar SSL com o Assistente de Serviços da Web e o WSRR.
 - Ao enviar o DFHLS2WS, o CICS gera o arquivo de ligação de serviço da web e o coloca no local que você especificou com o parâmetro **WSBIND**. A descrição do serviço da web gerada é colocada no local que você especificou com o parâmetro **WSDL**, **WSDL_1.1** ou **WSDL_2.0**.
 - Se você tiver usado os parâmetros WSRR no DFHLS2WS, seu documento WSDL será publicado no servidor WSRR que você especificou.
3. Revise a descrição de serviços da web gerada e execute qualquer customização necessária. Para obter mais informações, consulte “Customizando Documentos da Descrição do Serviço da Web Gerado” na página 582.
4. Copie o arquivo de ligação de serviço da web no diretório de recebimento do pipeline do modo de provedor que você deseja usar para seu aplicativo de serviço da web. Deve-se copiar o arquivo de ligação de serviço da web no modo binário.
5. Opcional: Copie a descrição de serviços da web ou o archive que contém um ou mais descrições de serviços da web no mesmo diretório que o arquivo de ligação de serviço da web. O archive deve ser um arquivo .zip e o nome do arquivo deve corresponder ao nome do arquivo WSDL. Com esta cópia, é possível descobrir o WSDL.
6. Use o comando **PIPELINE SCAN** para criar dinamicamente o recurso WEBSERVICE e dois recursos URIMAP. O recurso WEBSERVICE encapsula o arquivo de ligação de serviço da web no CICS e é usado no tempo de execução.
 - O primeiro recurso URIMAP fornece ao CICS as informações para associar o recurso WEBSERVICE a um URI específico.
 - O segundo recurso URIMAP fornece ao CICS as informações para associar o archive WSDL ou documento WSDL a um URI específico.
 - Este URI tem o mesmo caminho que o URI associado ao WEBSERVICE com o sufixo `?wsdl` anexado.

- Este recurso URIMAP é criado para que os solicitantes externos possam usar o URI para descobrir o archive WSDL ou o documento WSDL.
- Este recurso URIMAP é criado somente se a descrição de serviços da web ou o archive que contém uma ou mais descrições de serviços da web foi copiado no mesmo diretório que o arquivo de ligação de serviço da web.
- Se o diretório de recebimento contiver um archive WSDL e um documento WSDL, o URI retornará somente o WSDL no archive.
- Esta função está disponível somente para serviços da web instalados usando a operação de varredura de pipeline.

Como alternativa, é possível definir os recursos você mesmo, embora isso não seja recomendado.

Resultados

Quando você tiver criado os recursos do CICS com êxito, a criação de seu aplicativo do provedor de serviços estará concluída.

Se você tiver algum problema ao enviar o DFHLS2WS ou se os recursos não forem instalados corretamente, consulte Diagnosticando erros de implementação.

O que Fazer Depois

Disponibilize a descrição dos serviços da web para qualquer pessoa que precise desenvolver um solicitante de serviço da web que acessará seu serviço.

Criando um Documento de Descrição de Canal

Crie um documento de descrição de canal quando seu aplicativo do provedor de serviços utilizar uma interface de canal com vários contêineres.

Sobre Esta Tarefa

Use um editor XML para criar o documento de descrição de canal. O esquema para a descrição de canal é chamado de `channel.xsd` e está no diretório `/usr/lpp/cicsts/cicsts52/schemas/channel` (em que `/usr/lpp/cicsts/cicsts52` é o diretório de instalação padrão para arquivos CICS).

Procedimento

1. Crie um documento XML com um elemento `<channel>` e o namespace do canal CICS:

```
<channel name="myChannel"
xmlns="http://www.ibm.com/xmlns/prod/CICS/channel">
</channel>
```

2. Inclua um elemento `<container>` para cada contêiner que a interface do programa de aplicativo usa no canal. Deve-se usar os atributos de nome, tipo e uso para descrever cada contêiner. O exemplo a seguir mostra seis contêineres com valores de atributo diferentes:

```
<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>

```

O elemento de estrutura indica que o conteúdo está definido em uma estrutura de linguagem localizada em um membro do conjunto de dados particionados.

3. Salve o documento XML no z/OS UNIX.

Esquema do Canal

O documento de descrição de canal deve estar em conformidade com o esquema a seguir:

```

<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 um canal do CICS.
2. Este elemento representa um contêiner do CICS dentro do canal.

3. Uma estrutura pode ser usada somente com contêineres do modo 'bit'. O atributo 'location' indica o local de um arquivo que mapeia o conteúdo do contêiner. O atributo 'structure' pode ser usado em C e C++ para indicar o nome da estrutura.

O que Fazer Depois

Execute DFHLS2WS para criar os mapeamentos e o documento WSDL para o aplicativo do provedor de serviço da web. O DFHLS2WS coloca os mapeamentos para o canal no documento WSDL na ordem que os contêineres são especificados no documento de descrição de canal.

Customizando Documentos da Descrição do Serviço da Web Gerado

Os documentos da descrição do serviço da web (WSDL) que são gerados por DFHLS2WS contêm alguns conteúdos gerados automaticamente que podem ser apropriados mudar antes da publicação. A customização de documentos WSDL pode resultar na nova geração do arquivo de ligação de serviços da web e, em alguns casos, na gravação de um programa wrapper.

Sobre Esta Tarefa

Siga estas etapas para customizar os documentos de descrição do serviço da web gerado:

Procedimento

1. Se deseja anunciar o suporte para HTTPS ou comunicar-se usando o WebSphere MQ, use o parâmetro **URI** no DFHLS2WS para configurar um URI absoluto. Se você não tiver usado o parâmetro **URI**, deverá mudar os elementos <wsdl:service> e <wsdl:binding> no final do documento WSDL. O WSDL gerado inclui comentários para ajudá-lo a fazer essas mudanças. A mudança nesses elementos não requer que você gere novamente o arquivo de ligação de serviços da web.
2. Se desejar fornecer o local de rede de seu serviço da web, use o parâmetro **URI** no DFHLS2WS para configurar um URI absoluto. Se você não tiver usado o parâmetro **URI**, inclua os detalhes no soap:address no elemento wsdl:service.
 - a. Se você estiver usando um protocolo baseado em HTTP, substitua *my-server* pelo nome do host TCP/IP de sua região CICS e *my-port* pelo número da porta do recurso TCPIP SERVICE.
 - b. Se você estiver usando o WebSphere MQ como o protocolo de transporte, substitua *myQueue* pelo nome da fila apropriada.

É possível fazer essas mudanças sem exigir nenhuma mudança no arquivo de ligação de serviços da web.

Se você estiver mudando o nome da porta e o namespace sem gerar novamente o arquivo WSBIND, as informações de monitoramento poderão estar erradas no nível de tempo de execução 2.1 em diante.

3. Considere se os nomes gerados automaticamente no documento WSDL são apropriados para seus propósitos. É possível renomear esses valores:
 - O targetNamespace do documento WSDL
 - O targetNamespace dos esquemas XML dentro do documento WSDL
 - O nome <wsdl:portType>
 - O nome <wsdl:operation>

- O nome <wsdl:binding>
- O nome <wsdl:service>
- Os nomes dos campos nos esquemas XML no documento WSDL.

Esses valores fazem parte da interface programática na qual você codifica um programa cliente. Se os nomes gerados não forem suficientemente significativos, a manutenção de seu código do aplicativo poderá ser mais difícil durante um longo período de tempo. Use os parâmetros **REQUEST-NAMESPACE** e **RESPONSE-NAMESPACE** do DFHLS2WS para mudar o targetNamespace dos esquemas XML e o parâmetro **WSDL-NAMESPACE** para mudar o targetNamespace do documento WSDL.

Se você mudar qualquer um desses valores, deverá usar DFHWS2LS para gerar novamente o arquivo de ligação de serviços da web. As estruturas de linguagem que são produzidas não serão as mesmas que suas estruturas de linguagem existentes, mas serão compatíveis com seu aplicativo existente, portanto, nenhuma mudança no aplicativo é necessária. No entanto, você pode ignorar as novas estruturas de linguagem e usar o novo arquivo de ligação de serviços da web com as estruturas originais.

4. Considere se os campos de COMMAREA expostos nos esquemas XML são apropriados. Você pode considerar remover quaisquer campos que não são úteis para um desenvolvedor de cliente de serviço da web:

- Os campos que são usados somente para valores de saída podem ser removidos do esquema que mapeia as estruturas de dados de entrada.
- Campos de preenchimento.
- Anotações geradas automaticamente.

Se você fizer alguma dessas mudanças, deverá gerar novamente o arquivo de ligação de serviços da web usando DFHWS2LS. As novas estruturas de linguagem que são geradas não são compatíveis com as estruturas de linguagem originais, portanto, você deve gravar um programa wrapper para mapear dados da nova representação para a antiga. Este programa wrapper precisa executar um comando **EXEC CICS LINK** para o programa de aplicativo de destino e, em seguida, mapear os dados retornados.

Este nível de customização requer o maior esforço, mas resulta em interfaces programáticas mais significativas para seus desenvolvedores de cliente de serviços da web.

5. Se desejar colocar o documento WSDL gerado por meio do DFHWS2LS para criar novas estruturas de linguagem, decida se deve manter as anotações no documento WSDL. As anotações substituem as regras de mapeamento normais quando o DFHWS2LS gera as estruturas de linguagem. Ao substituir as regras de mapeamento, assegure que as estruturas de linguagem geradas sejam compatíveis com a versão que foi usada pelo DFHLS2WS. Se desejar usar as regras de mapeamento padrão para produzir as estruturas de linguagem, remova as anotações.

Resultados

Se desejar publicar seu documento WSDL customizado em um servidor IBM WebSphere Service Registry and Repository (WSRR), você deverá publicá-lo manualmente usando a interface do WSRR. É possível localizar mais informações sobre WSRR no local a seguir: .

Exemplo

Para obter um exemplo de um documento WSDL, consulte Um Exemplo do Documento WSDL Gerado.

Enviando uma Falha de SOAP

Em um provedor de serviços, é possível usar a API do CICS para enviar uma falha de SOAP para um solicitante de serviço da web. A falha pode ser emitida pelo aplicativo do provedor de serviços ou por um programa de processamento de cabeçalho no pipeline.

Antes de Iniciar

Para usar a API, o aplicativo do provedor de serviços deve usar canais e contêineres. Se o aplicativo usar COMMAREAs, grave um programa wrapper que usa canais e contêineres para criar a mensagem de falha de SOAP. É possível usar a API em um programa de processamento de cabeçalho somente se ela é chamada diretamente a partir de um manipulador de mensagem SOAP fornecido pelo CICS.

Sobre Esta Tarefa

Você pode desejar emitir uma falha de SOAP para o solicitante de serviço da web se sua lógica de aplicativo não pode satisfazer a solicitação, por exemplo, ou se há um problema subjacente com a mensagem de solicitação. Observe que o CICS não considera emitir uma falha de SOAP como um erro, portanto, o processamento de pipeline de resposta da mensagem normal ocorre no lugar de qualquer processamento de erro. Se você não deseja recuperar quaisquer transações, deve usar o programa de aplicativo.

Procedimento

1. Em seu programa, use o comando **EXEC CICS SOAPFAULT CREATE** para enviar uma falha de SOAP.
2. Inclua a opção **CLIENT** ou **SERVER** no comando. Esta opção indica onde o problema ocorreu, no lado do cliente ou no servidor.
 - **CLIENT** indica que o problema é com a mensagem de solicitação que foi recebida.
 - **SERVER** indica que o problema ocorre quando a mensagem de solicitação é processada pelo CICS. Esse problema pode estar em um programa de aplicativo, por exemplo, ele pode ser incapaz de satisfazer a solicitação ou pode ser um problema subjacente que ocorre durante o processamento de pipeline.
3. Inclua a opção **FAULTSTRING** e seu comprimento na opção **FAULTSTRLEN** para fornecer um resumo do motivo pelo qual a falha foi emitida pelo provedor de serviços. O conteúdo desta opção está em XML. Quaisquer dados fornecidos pelo aplicativo devem estar em um formato que seja adequado para inclusão direta em um documento XML. O aplicativo pode precisar especificar alguns caracteres como entidades XML. Por exemplo, se o caractere `<` for usado em qualquer lugar diferente do início de uma tag XML, o aplicativo deverá alterá-lo para `<`. O exemplo a seguir mostra uma opção **FAULTSTRING** incorreta:

```
dcl msg_faultString char(*) constant('Error: Value A
< Value B');
```

A maneira correta de especificar esta opção **FAULTSTRING** é a seguinte:


```
dc1 msg_faultString char(*) constant('Error: Value A
&lt; Value B');
```

Dica: Para evitar usar entidades XML, é possível agrupar os dados em uma construção XML CDATA. Os processadores XML não analisam dados de caractere nesta construção. Usando este método, você poderia especificar a opção FAULTSTRING a seguir:

```
dc1 msg_faultString char(*)
constant('<![CDATA[Error: Value A < Value B]]>');
```

4. Codifique a opção DETAIL e seu comprimento na opção DETAILLENGTH para fornecer os detalhes do motivo pelo qual a falha foi emitida pelo provedor de serviços. O conteúdo desta opção está em XML. A mesma orientação aplicada na opção DETAIL se aplica na opção FAULTSTRING.
5. Opcional: Se você estiver chamando a API a partir de um programa de processamento de cabeçalho, defina o programa no arquivo de configuração de pipeline. O programa de processamento de cabeçalho é definido no elemento `<cics_soap_1.1_handler>`, `<cics_soap_1.2_handler>`, `<cics_soap_1.1_handler_java>` ou `<cics_soap_1.2_handler_java>`.

Resultados

Quando seu programa emite esse comando, o CICS cria a mensagem de resposta de falha SOAP no nível SOAP apropriado. Se seu aplicativo do provedor de serviços emite o comando, ele não precisa criar uma resposta SOAP e colocá-la no contêiner DFHRESPONSE. O pipeline processa a falha de SOAP por meio dos manipuladores de mensagem e envia a resposta para o provedor de serviços da web.

Exemplo

O comando **SOAPFAULT CREATE** tem várias opções para fornecer-lhe flexibilidade para responder apropriadamente a um solicitante de serviço da web. A seguir há um exemplo simples de um comando concluído que cria uma falha de SOAP que pode ser usado para SOAP 1.1 e SOAP 1.2:

```
EXEC CICS SOAPFAULT CREATE CLIENT
DETAIL(
msg_detail
)
DETAILLENGTH(length(
msg_detail
))
FAULTSTRING(
msg_faultString
)
FAULTSTRLEN(length(
msg_faultString
));
```

É possível codificar *msg_detail* e *msg_faultString* com os valores a seguir:

```
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');
```

Criando um solicitante de serviço da web usando o assistente de serviços da web

É possível criar um aplicativo do solicitante de serviço a partir de uma descrição de serviço da web que está em conformidade com WSDL 1.1 ou WSDL 2.0. O assistente de serviços da web do CICS ajuda a implementar seus aplicativos CICS em uma configuração do solicitante de serviço.

Antes de Iniciar

A descrição de serviços da web deve estar em um arquivo no z/OS UNIX ou deve ser publicada em um servidor IBM WebSphere Services Registry and Repository (WSRR) e um pipeline no modo do solicitante deve ser instalado na região CICS.

Deve-se alocar armazenamento suficiente para o ID do usuário para que o ID possa executar Java. É possível usar qualquer versão suportada de Java. Por padrão, DFHWS2LS usa a versão de Java especificada no parâmetro **JAVADIR**.

Sobre Esta Tarefa

Ao usar o assistente de serviço da web do CICS para implementar um aplicativo CICS como um solicitante de serviços, deve-se iniciar com uma descrição do serviço da web e gerar as estruturas de dados de linguagem a partir dela.

Procedimento

1. Use o programa em lote DFHWS2LS para gerar um arquivo de ligação de serviço da web e uma ou mais estruturas de linguagem. Considere estas opções ao criar um aplicativo do solicitante de serviço a partir de uma descrição de serviços da web:
 - Qual nível de mapeamento deseja usar? Quanto maior o nível de mapeamento, mais controle e apoio você tem disponíveis para a manipulação de dados de caracteres e binários no tempo de execução. Alguns parâmetros opcionais estão disponíveis somente nos níveis de mapeamento mais altos. É recomendável usar o nível mais alto de mapeamento disponível.
 - Qual página de códigos você deseja usar ao transformar dados no tempo de execução? Se você deseja usar uma página de códigos específica para seu aplicativo que seja diferente da página de códigos para a região CICS, use o parâmetro **CCSID**.
 - Você deseja suportar um subconjunto das operações que são declaradas na descrição do serviço da web? Se você tiver uma descrição de serviços da web muito grande e desejar que o aplicativo do solicitante de serviço suporte somente um determinado número de operações, use o parâmetro **OPERATION** para listar as que você deseja. Cada operação deve ser separada por um espaço e faz distinção entre maiúsculas e minúsculas.
 - Onde o documento WSDL é armazenado? Se o documento WSDL que você deseja usar como entrada para DFHWS2LS estiver armazenado em um servidor WSRR, será possível recuperá-lo executando DFHWS2LS com determinados parâmetros especificados. Use o parâmetro **WSRR-SERVER** para especificar o local do servidor WSRR e use o parâmetro **WSRR-NAME** para especificar o nome do documento WSDL que você deseja recuperar. Para obter informações sobre outros parâmetros no DFHWS2LS que você pode desejar usar para interagir com o WSRR, consulte “DFHWS2LS: WSDL para Conversão de Linguagem de Alto Nível” na página 504.

- Se deseja recuperar o documento WSDL a partir de um servidor WSRR, você quer fazer isso usando uma conexão segura? É possível usar a criptografia secure socket layer (SSL) com o assistente de serviços da web para interoperar com segurança com o WSRR. Para obter um exemplo, consulte Exemplo de Como Usar SSL com o Assistente de Serviços da Web e o WSRR.

Não especifique parâmetros como **PROGRAM**, **URI**, **TRANSACTION** e **USERID** quando usar DFHWS2LS. Esses parâmetros se aplicam somente a um aplicativo do provedor de serviços e, se especificados, fazem com que um arquivo de ligação de serviços da web no modo de provedor seja produzido. Além do arquivo de ligação de serviço da web, o programa gera uma estrutura de dados de linguagem.

2. Verifique o arquivo de log para ver se ocorreu algum problema quando o DFHWS2LS gerou o arquivo de ligação e as estruturas de linguagem. O CICS pode não suportar alguns elementos ou opções na descrição do serviço da web. Se quaisquer mensagens de aviso ou de erro forem emitidas, leia o conselho que é fornecido e execute a ação apropriada. Pode ser necessário executar novamente o programa em lote.
3. Copie o arquivo de ligação de serviço da web no diretório de recebimento do pipeline no modo do solicitante que você deseja usar para seu aplicativo de serviço da web.
4. Assegure que o recurso PIPELINE esteja configurado para aplicativos do solicitante de serviço. O valor do parâmetro **MODE** mostra se o pipeline suporta aplicativos de serviços da web de solicitante ou provedor.
5. Assegure que o protocolo SOAP correto seja suportado no pipeline para seu serviço da web. O parâmetro **SOAPLEVEL** indica qual versão é suportada. No modo do solicitante de serviço, a ligação do serviço da web deve corresponder à versão de SOAP que é suportada no pipeline. Não é possível instalar um serviço da web com uma ligação SOAP 1.1 em um pipeline do solicitante de serviço que suporta SOAP 1.2.
6. Assegure que o tempo limite configurado para o pipeline seja adequado para o aplicativo do solicitante de serviço. O tempo limite é exibido como o valor do atributo RESPWAIT no recurso PIPELINE. Se nenhum limite for especificado no pipeline, o padrão para o transporte será utilizado.
 - O tempo limite padrão para HTTP é 10 segundos.
 - O tempo limite padrão para o WebSphere MQ é 60 segundos.

Cada transação na região CICS tem um tempo limite do dispatcher. Se esse valor for menor que o padrão para qualquer protocolo, o tempo limite ocorrerá com o dispatcher.

7. Opcional: Copie a descrição de serviços da web no mesmo diretório de recebimento que o arquivo de ligação de serviço da web, para que você possa ativar a validação para o serviço da web no tempo de execução.
8. Crie o recurso WEBSERVICE. Este recurso contém o arquivo de ligação de serviço da web no CICS e é usado no tempo de execução.

Isso pode ser feito das seguintes maneiras:

- a. Usando o comando **PIPELINE SCAN** para criar dinamicamente o recurso WEBSERVICE.
- b. Definindo o recurso você mesmo. Se você usar o CICS Explorer para definir um recurso WEBSERVICE em um pacote configurável do CICS, poderá optar por importar um arquivo de ligação de serviço da web e um documento WSDL ou archive WSDL e incluí-los no pacote configurável. É possível, então, gerar definições de URIMAP para suportar o serviço da

web e empacotá-las em um pacote configurável. Para obter mais ajuda com o uso do CICS Explorer para criar e editar recursos nos pacotes configuráveis do CICS, consulte Trabalhando com pacotes configuráveis na documentação do produto CICS Explorer.

9. Grave um programa wrapper que você possa substituir por sua lógica de comunicações. Use a estrutura de dados de linguagem gerada na etapa 1 para gravar seu programa wrapper. Use um comando **EXEC CICS INVOKE SERVICE** em seu programa wrapper para se comunicar com o serviço da web. O comando inclui estas opções:

- O URI do serviço da web
- A operação para a qual o serviço da web está sendo chamado

Ao chamar o serviço da web, você pode especificar um recurso URIMAP que contém as informações sobre o URI do serviço da web. É possível especificar estas informações diretamente no comando INVOKE SERVICE em vez de usar um recurso URIMAP. Entretanto, usar um recurso URIMAP significa que você não precisa recompilar seus aplicativos se o URI de um provedor de serviços for alterado. Com um recurso URIMAP, também é possível escolher para implementar a definição do conjunto de conexões, na qual o CICS mantém a conexão do cliente aberta após o uso, para que ela possa ser reutilizada pelo aplicativo para solicitações subsequentes ou por um outro aplicativo que chama o mesmo serviço. O comando PIPELINE SCAN não cria recursos URIMAP para um solicitante de serviços, portanto, você mesmo deve definir o recurso URIMAP seguindo as instruções em Criando um recurso URIMAP para CICS como um cliente HTTP.

Resultados

Quando você tiver criado os recursos do CICS com êxito, a criação de seu aplicativo do solicitante de serviço será concluída.

Criando um Serviço da Web Usando o Conjunto de Ferramentas

Em vez de usar o assistente de serviços da web JCL, é possível usar o IBM Developer for z Systems ou gravar seu próprio programa Java para criar os arquivos necessários no CICS.

Procedimento

1. Há duas opções:
 - Usar a ferramenta IBM Developer for z Systems para criar um arquivo de ligação de serviço da web e a descrição de serviços da web ou estruturas de linguagem. Para obter mais informações sobre esta ferramenta, consulte .
 - Grave seu próprio programa Java usando a API fornecida para chamar o assistente de serviços da web. Esta API é descrita no Javadoc Assistente de Serviços da Web: Referência de Classe. Ela inclui comentários que explicam as classes e é fornecido um código de amostra para fornecer um exemplo de como você pode chamar o assistente de serviços da web. O Javadoc também contém uma lista completa dos arquivos JAR que são necessários e de seus locais no z/OS UNIX.

É possível executar seu programa Java na plataforma z/OS. Windows ou Linux. Se você executar o programa no Windows ou Linux, transfira o arquivo de ligação de serviços da web gerado para um diretório de recebimento adequado no modo binário usando FTP ou um processo equivalente.

2. Opcional: Se você estiver gerando uma descrição de serviços da web a partir de uma estrutura de linguagem, revise o arquivo e execute qualquer customização necessária. Para obter mais informações, consulte “Customizando Documentos da Descrição do Serviço da Web Gerado” na página 582.
3. Implemente o arquivo de ligação de serviço da web gerado em um diretório de recebimento de pipeline adequado.
4. Opcional: Copie a descrição de serviços da web no diretório de recebimento do pipeline, para que você possa executar a validação do serviço da web para verificar se ele está funcionando conforme o esperado.
5. Se você iniciou com uma descrição de serviços da web, grave um provedor de serviços ou programa do aplicativo solicitante para criar uma interface com as estruturas de linguagem geradas.
6. Execute um comando **PIPELINE SCAN** para criar automaticamente os recursos necessários do CICS.

Criando Seus Próprios Aplicativos de Serviço da Web que Reconhecem XML

Se você decidir não utilizar os mapeamentos de dados fornecidos pelo CICS, poderá gravar seus próprios aplicativos de dados cientes do XML de duas maneiras. É possível usar o parâmetro **XML-ONLY** no DFHWS2LS ou você pode gravar seu próprio aplicativo sem usar qualquer ferramenta. Usar o parâmetro **XML-ONLY** é a maneira mais direta de configurar o processo de pipeline do CICS para transmitir os dados XML ao aplicativo para que sejam manipulados.

Sobre Esta Tarefa

A gravação de seus próprios aplicativos que reconhecem XML envolve gravar o código para analisar e gerar documentos XML. Uma maneira de gravar seu próprio aplicativo que reconhece XML usa as instruções XML PARSE e XML GENERATE em COBOL. Outra maneira de gravar seus próprios aplicativos que reconhecem XML usa outras ferramentas IBM; por exemplo, é possível usar o IBM Developer for z Systems para ferramenta para gerar programas conversores COBOL XML que podem ser chamados a partir de seus aplicativos.

Criando um Aplicativo do Provedor de Serviços que Reconhece XML

Seu aplicativo do provedor de serviços que reconhece XML deve trabalhar com os contêineres que são transmitidos a ele e manipular a conversão de dados entre o XML e a linguagem do programa.

Sobre Esta Tarefa

As etapas a seguir orientam você através da criação de seu aplicativo que reconhece XML, incluindo a decisão sobre o uso de qualquer um dos conjuntos de ferramentas do CICS.

Procedimento

1. Decida se deseja gerar um arquivo de ligação de serviço da web para seu aplicativo que reconhece XML usando DFHWS2LS. A vantagem de gerar um arquivo de ligação de serviço da web é que você pode usar serviços CICS, tal como a validação, para testar seu serviço da web e o monitoramento do CICS usando saídas de usuário global.

- Se você deseja gerar um arquivo de ligação de serviço da web, execute DFHWS2LS especificando o parâmetro **XML-ONLY** e um **MINIMUM-RUNTIME-LEVEL** de 2.1 ou superior. O arquivo de ligação de serviço da web permite que o programa de aplicativo trabalhe diretamente com o conteúdo do contêiner DFHWS-BODY. Em todos os outros aspectos, o arquivo de ligação gerado compartilha as mesmas características de implementação e o mesmo comportamento de tempo de execução que um arquivo gerado sem o parâmetro **XML-ONLY**, incluindo a análise do XML durante a manipulação da mensagem SOAP. Para evitar esta análise, você não deve especificar “Os Manipuladores de Mensagem SOAP” na página 143 em seu arquivo de configuração de pipeline.
 - Se você não deseja usar um arquivo de ligação de serviço da web, configure seu pipeline do provedor de serviços para que a solicitação de serviço da web atinja seu aplicativo que reconhece XML. É possível configurar o manipulador de terminal no arquivo de configuração de pipeline para usar seu programa de aplicativo que reconhece XML ou criar um manipulador de mensagem que alterna dinamicamente para seu aplicativo dependendo do URI que é recebido no pipeline.
2. Grave seu aplicativo para manipular a solicitação de serviço da web que é mantida nos contêineres a seguir:

DFHWS-BODY

O conteúdo do corpo SOAP para uma solicitação SOAP de entrada quando o pipeline inclui um manipulador de mensagem SOAP fornecido pelo CICS.

DFHREQUEST

A solicitação completa, incluindo o envelope para uma solicitação SOAP, recebida a partir do pipeline.

DFHWS-XMLNS

Uma lista de pares nome-valor que mapeiam prefixos de namespace para namespaces para o conteúdo XML da solicitação.

DFHWS-SOAPACTION

O cabeçalho SOAPAction associado à mensagem SOAP no contêiner DFHWS-BODY.

Ao codificar comandos de API para trabalhar com os contêineres, não especifique a opção CHANNEL, porque todos os contêineres são associados ao canal atual (o canal que foi transmitido ao programa). Se precisar saber o nome do canal, use o comando **EXEC CICS ASSIGN CHANNEL**.

3. Opcional: Seu aplicativo também pode usar contêineres adicionais que estão disponíveis para manipuladores de mensagem no pipeline, bem como quaisquer outros contêineres que os manipuladores de mensagens criam como parte de seu processamento. Para obter uma lista completa de contêineres, consulte “Contêineres usados no pipeline” na página 148.
4. Quando seu aplicativo tiver processado a solicitação, construa uma resposta de serviço da web usando os contêineres a seguir:

DFHRESPONSE

A mensagem de resposta completa a ser transmitida ao pipeline. Use este contêiner se você não usar SOAP para suas mensagens ou se você desejar construir a mensagem SOAP completa, incluindo o envelope, em seu programa em vez de usar o manipulador de mensagem SOAP fornecido pelo CICS.

Se você fornecer um corpo SOAP no contêiner DFHWS-BODY, DFHRESPONSE será ignorado.

DFHWS-BODY

Para obter uma resposta SOAP de saída, o conteúdo do corpo SOAP. Forneça este contêiner quando o manipulador de terminal de seu pipeline for um manipulador de mensagem SOAP fornecido pelo CICS. O manipulador de mensagem constrói a mensagem SOAP completa contendo o corpo.

Seu programa deverá criar este contêiner, mesmo se a solicitação e a resposta forem idênticas. Se você não o fizer, o CICS emitirá um erro interno do servidor.

Também é possível usar qualquer um dos outros contêineres para transmitir informações de que seu pipeline precisa para processar a resposta de saída.

Se o seu serviço da web não retornar uma resposta, você deverá retornar o contêiner DFHNORESPONSE para indicar que não há resposta. O conteúdo do contêiner não é importante, porque o manipulador de mensagens verifica somente se o contêiner está presente ou não.

5. Crie um recurso URIMAP. Se você estiver usando o parâmetro **XML-ONLY** e tiver especificado um valor para o parâmetro **URI** de DFHWS2LS, o URIMAP será criado automaticamente para você durante o processo PIPELINE SCAN.

Criando um Aplicativo do Solicitante de Serviço que Reconhece XML

Seu aplicativo do solicitante de serviço da web que reconhece o XML manipula a conversão de dados entre o XML e a linguagem de programação e preenche os contêineres de controle no pipeline.

Antes de Iniciar

É possível gravar seu próprio aplicativo do solicitante de serviço que reconhece XML usando o parâmetro **XML-ONLY** em DFHWS2LS ou você pode gravá-lo sem usar qualquer ferramenta. A maneira mais direta de gravar seu próprio aplicativo do solicitante de serviço que reconhece XML é usando o parâmetro **XML-ONLY** em DFHWS2LS; o parâmetro **XML-ONLY** está disponível no nível de tempo de execução 2.1 e posterior.

Sobre Esta Tarefa

O uso do parâmetro **XML-ONLY** resulta na geração de um arquivo WSBIND que informa ao CICS que o aplicativo trabalhará diretamente com o conteúdo do contêiner DFHWS-BODY. O arquivo WSBIND gerado deve ser instalado em um PIPELINE no modo de solicitante para criar um recurso WEBSERVICE no modo de solicitante. O aplicativo deve gerar XML para o corpo da solicitação de serviço da web e armazená-lo no contêiner DFHWS-BODY. Ele deve, então, chamar o comando **EXEC CICS INVOKE SERVICE**. A mensagem de saída é enviada ao provedor de serviços da web. O corpo da mensagem de resposta também está no contêiner DFHWS-BODY após a chamada ser concluída.

O XML das mensagens de resposta é analisado durante a manipulação da mensagem SOAP. Para evitar esta análise, você não deve especificar Manipuladores de mensagem SOAP em seu arquivo de configuração de pipeline.

Os aplicativos do solicitante cientes do XML podem receber mensagens de Falha de SOAP de volta do aplicativo no modo de provedor remoto. Nesse caso, o aplicativo do solicitante é responsável por interpretar a Falha de SOAP e distingui-la de uma mensagem de resposta regular. Se o comando **INVOKE SERVICE**

for usado com um WEBSERVICE **XML-ONLY**, o CICS não configurará o código de resposta que normalmente é usado para indicar que uma Falha de SOAP foi recebida.

Se você estiver gravando seu próprio aplicativo do solicitante de serviço que reconhece XML sem usar a opção **XML-ONLY**, conclua as etapas a seguir:

Procedimento

1. Crie um canal e o preencha com contêineres. Os contêineres de controle devem ser preenchidos no modo CHAR. Forneça as informações a seguir em cada contêiner:

DFHWS-PIPELINE

O nome do recurso PIPELINE usado para a solicitação de saída.

DFHWS-URI

O URI do serviço da web de destino

DFHWS-BODY

Para uma solicitação SOAP de saída, o conteúdo do corpo SOAP. Forneça este contêiner quando o pipeline incluir um manipulador de mensagem SOAP fornecido pelo CICS. O manipulador de mensagem constrói a mensagem SOAP completa contendo o corpo.

DFHREQUEST

A mensagem de solicitação completa a ser transmitida ao pipeline. Use este contêiner se você não usar SOAP para suas mensagens ou se desejar construir a mensagem SOAP completa, incluindo o envelope, em seu programa. O pipeline não deve incluir um manipulador de mensagem SOAP fornecido pelo CICS para evitar que cabeçalhos SOAP duplicados sejam enviados na mensagem de saída.

Se você fornecer um corpo SOAP no contêiner DFHWS-BODY, DFHREQUEST deverá estar vazio. Se você fornecer conteúdo em DFHWS-BODY e DFHREQUEST, o CICS usará DFHREQUEST.

DFHWS-XMLNS

Uma lista de pares nome-valor que mapeiam prefixos de namespace para namespaces para o conteúdo XML da solicitação.

DFHWS-SOAPACTION

O cabeçalho SOAPAction a ser incluído na mensagem SOAP especificada no contêiner DFHWS-BODY.

Dica: Se você incluir o contêiner DFHWS-NOABEND no canal, quando DFHPIRT for chamado, nenhum encerramento anormal será emitido a partir de DFHPIRT. Isto é útil se você está executando um programa C/C++, porque é possível manipular erros por meio do contêiner DFHERROR.

2. Vincule ao programa DFHPIRT. Use este comando:

```
EXEC CICS LINK PROGRAM(DFHPIRT) CHANNEL(  
  userchannel  
)
```

em que *userchannel* é o canal que contém seus contêineres. A mensagem de saída é processada pelos manipuladores de mensagem e programas de processamento de cabeçalho no pipeline e enviada ao provedor de serviços da web.

3. Recupere os contêineres que contêm a resposta do serviço da web do mesmo canal. A resposta do provedor de serviços da web pode ser uma resposta bem-sucedida ou uma falha de SOAP. O aplicativo do solicitante de serviço da web deve ser capaz de manipular ambos os tipos de resposta do provedor de serviços. A resposta completa está contida nos contêineres a seguir:

DFHRESPONSE

A resposta completa, incluindo o envelope para uma resposta SOAP, recebida do provedor de serviços da web.

DFHWS-BODY

Quando o pipeline inclui um manipulador de mensagem SOAP fornecido pelo CICS, o conteúdo do corpo SOAP.

DFHERROR

Informações de erro do pipeline.

Nota: Em alguns casos de erro o DFHWS-BODY não pode ser atualizado. Deve-se verificar o DFHRESPONSE em busca de uma falha de SOAP.

Usando Java com serviços da web

É possível usar Java para criar aplicativos de serviço da web. Diferentes técnicas são usadas para criar esses aplicativos comparados com as técnicas usadas com outras linguagens de programação.

Para a maioria das linguagens de programação não Java, você usa os assistentes de serviços da web para ativar aplicativos. O uso do assistente de serviços da web significa que o CICS converterá os dados do serviço da web em um formato adequado para o aplicativo e o colocará em um contêiner ou COMMAREA. É possível usar o assistente de serviços da web com os aplicativos Java, no entanto, as tarefas a seguir fornecem métodos mais adequados para criar serviços da web Java para aplicativos Java.

Implementando um serviço da web no modo de provedor Java em um servidor JVM Axis2

É possível implementar um aplicativo Axis2 como um serviço da web no modo de provedor no CICS. Esses aplicativos geralmente são gerados usando JAX-WS e podem ser hospedados em um pipeline ativado por Java.

Você pode desejar implementar os aplicativos Java usando esse método por um dos motivos a seguir:

- Você tem investimento existente usando interfaces do Manipulador Axis2.
- Você deseja usar uma configuração de pipeline do CICS.

Nota: Aplicativos no estilo Axis2 não usam os recursos WEBSERVICE. Eles interagem com o CICS usando o modelo de programação Axis2 e, portanto, não podem usar alguns dos suportes de serviços da web do CICS. Os serviços a seguir não são totalmente suportados para aplicativos no estilo Axis2:

- SOAPFAULT CREATE
- WSACONTEXT GET
- “Contêiner DFHWS-OPERATION” na página 163
- “Contêiner DFHWS-MEP” na página 162
- “Contêiner DFHWS-USERID” na página 168

- “Contêiner DFHWS-TRANID” na página 164
- Segurança de Serviços do Web

Antes de Iniciar

Deve-se ter um aplicativo Java que seja adequado para implementação em Axis2, por exemplo, um aplicativo POJO usando JAX-WS. Para esta tarefa, o aplicativo POJO a seguir é usado como um exemplo:

```
/**
 * 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 + "'";
    }
}
```

Este aplicativo especifica o namespace XML que é usado para gerar o WSDL e um nome para associar ao serviço da web.

O código Java para este aplicativo deve ser compilado e o gerador de JAX-WS executado para empacotar o aplicativo em um arquivo jar chamado TestAxis2.jar. É possível fazer isto emitindo o código a seguir:

```
javac TestAxis2.java
wsген -cp . TestAxis2 -wsdl
jar -cvf TestAxis2.jar *
```

O gerador de JAX-WS também cria um documento WSDL e as ligações usadas por Axis2.

Sobre Esta Tarefa

Para implementar um serviço da web Axis2, deve-se criar a infraestrutura de pipeline para seus serviços da web. Quando você tiver criado o pipeline, poderá criar seus serviços da web. É possível reutilizar o pipeline criado para tantos serviços da web quanto necessários. As etapas a seguir descrevem como criar o pipeline e os serviços da web.

Nota: Nenhum recurso WEBSERVICE é criado ou instalado como parte desta tarefa.

Procedimento

1. Crie a infraestrutura do pipeline.
 - a. Crie uma infraestrutura de serviço da web para um pipeline Java. Para obter mais informações, consulte “Criando a infraestrutura do CICS para um provedor de serviços SOAP” na página 62.
 - b. Crie um repositório Axis2. Para fazer isso, crie uma cópia do repositório fornecido localizado em \$CICS_HOME/lib/pipeline/repository.
 - c. Inclua o elemento <repository> em seu arquivo de configuração de pipeline. Esse elemento deve especificar o nome do repositório Axis2 que você criou.
 - d. Crie e ative um recurso PIPELINE.

2. Para cada serviço da web associado ao pipeline, repita as etapas a seguir para criar o serviço da web.
 - a. Implemente o aplicativo Axis2 no repositório Axis2. Por exemplo, o arquivo jar criado no exemplo deve ser implementado em um diretório chamado `servicejars` no diretório do repositório. Deve-se criar esse diretório se ele não existir.
 - b. Defina e instale um recurso URIMAP para o serviço da web. O recurso URIMAP deve especificar o URI e o recurso PIPELINE associado ao serviço da web. O URI deve seguir as convenções de nomenclatura Axis2 para URIs. A convenção de nomenclatura do Axis2 padrão é: `/ Serviço name_of_service. Porta name_of_port/ suffix`, em que *name_of_service* é o nome do serviço da web no WSDL, *name_of_port* é o nome da porta no WSDL e *suffix* é um sufixo opcional que você pode definir. Para o exemplo anterior, o recurso URIMAP a seguir poderia ser usado:

```

Urimap : EXAMPLE
Group : EXAMPLE
SStatus : Enabled
USAge : Pipeline
SCheme : HTTP
P0rt : No
HOST : *
PAtH : /TestAxis2Service.pojoExamplePort/example/TestAxis2
TRansaction : CPIH
PIpeline : EXAMPLE

```

Este exemplo assume que o recurso PIPELINE usado é chamado de EXAMPLE.

O que Fazer Depois

Teste se seus serviços da web são executados corretamente.

Criando um serviço da web Java que gera e analisa XML

É possível criar aplicativos Java que analisam e geram XML eles mesmos. Esses aplicativos são consistentes com aplicativos que reconhecem XML gravados em outras linguagens de programação, mas eles se beneficiam do uso de tecnologias Java padrão para processar o XML.

Procedimento

1. Crie um recurso XML-ONLY WEBSERVICE. Para obter mais informações, consulte “Criando um Aplicativo do Solicitante de Serviço que Reconhece XML” na página 591 ou “Criando um Aplicativo do Provedor de Serviços que Reconhece XML” na página 589.
2. Grave um serviço da web Java que possa analisar e gerar XML para o corpo da mensagem SOAP. É possível usar várias ferramentas, como a biblioteca Java 6 Java Architecture for XML Binding (JAXB) para ajudá-lo a criar um serviço da web Java com esses recursos.
3. Opcional: Se você estiver usando um pipeline do provedor e desejar incluir o recurso para que uma mensagem de Falha de SOAP seja retornada ao solicitante, use a classe JCICS SoapFault para emitir o comando **EXEC CICS SOAPFAULT CREATE**.
4. Opcional: Se você estiver usando um pipeline do solicitante, use a classe de Serviço JCICS para emitir o comando **EXEC CICS INVOKE SERVICE**.

Criando um serviço da web Java que possui uma interface COBOL

É possível criar aplicativos Java que interagem com o CICS usando as mesmas técnicas usadas em outras linguagens de programação. Para criar esses aplicativos, deve-se gravar ou gerar o código Java para criar dados estruturados no estilo de COMMAERA ou contêiner.

Procedimento

1. Use DFHWS2LS para criar estruturas de linguagem COBOL para o serviço da web.
2. Grave um serviço da web Java que gera e analisa estruturas de linguagem COBOL. Para obter mais informações sobre ferramentas que permitem que os programas Java acessem os dados do aplicativo CICS existentes e links para exemplos de como criar um serviço da web Java que pode gerar e analisar estruturas de linguagem COBOL, consulte Interagindo com Dados Estruturados de Java.
3. Opcional: Se você estiver usando um pipeline do provedor e desejar incluir o recurso para que uma mensagem de Falha de SOAP seja retornada ao solicitante, use a classe JCICS SoapFault para emitir o comando **EXEC CICS SOAPFAULT CREATE**.
4. Opcional: Se você estiver usando um pipeline do solicitante, use a classe de Serviço JCICS para criar interface com a CICS SERVICE API e emita o comando **EXEC CICS INVOKE SERVICE**.

Implementando um serviço da Web JAX-WS no modo do solicitante

É possível implementar um aplicativo JAX-WS como um serviço da web no modo do solicitante no CICS. No entanto, esses aplicativos não usam o comando **EXEC CICS INVOKE**, em vez disso eles interagem com os serviços da web remotos usando JAX-WS.

Antes de Iniciar

Deve-se ter um servidor JVM configurado para suportar OSGi. Para obter mais informações, consulte Configurando um Servidor JVM.

Sobre Esta Tarefa

A vantagem de implementar um aplicativo JAX-WS como um serviço da web no modo do solicitante é que você cria um aplicativo do solicitante de serviço da web independente, que usa o zEnterprise Application Assist Processor (zAAP). O uso de zAAP pode reduzir o custo de transações; para obter mais informações, consulte a publicação IBM Redbooks: Implementação do zSeries Application Assist Processor (zAAP).

Procedimento

1. Crie um aplicativo do solicitante de serviço da web no Java e use uma API apropriada, tal como os serviços da web Java API for XML (JAX-WS), para chamar o serviço da web remoto.
2. Opcional: Se você usar o JAX-WS para iniciar um serviço da web remoto, também deverá usar o JAX-WS para gerar as mensagens SOAP, manipular a comunicação de rede e processar a resposta SOAP.
3. Implemente seu aplicativo Java e instale-o no servidor JVM.

O que Fazer Depois

Teste se seus serviços da web iniciam corretamente.

Implementando um serviço da web no modo de provedor Java em um servidor Liberty JVM

É possível implementar um aplicativo da web como um serviço da web no modo de provedor em um servidor Liberty JVM. Esses aplicativos são criados usando os padrões Java JAX-WS e JAXB. Este tópico se aplica somente ao Liberty de modo integrado do CICS.

Sobre Esta Tarefa

O CICS TS V5.3 inclui o WebSphere Application Server Liberty Profile (WLP) mais recente que fornece recursos para o Java API for XML Web Services (JAX-WS) e o Java Architecture for XML Binding (JAXB). Juntas essas tecnologias permitem que você grave serviços da web SOAP no Java como parte de um aplicativo CICS. O artigo a seguir mostrará como configurar o Eclipse, testar um projeto de serviços da web de amostra, implementar a amostra no CICS, modificar a amostra para usar JCICS e testá-la com o Explorador de Serviços da Web. Para obter mais informações, consulte o artigo do CICS DevCenter, Amostra de serviço da web JAX-WS para Liberty.

Você pode desejar implementar os aplicativos Java usando esse método por um dos motivos a seguir:

- Você deseja criar os serviços da web em Java.
- Você tem documentos WSDL complicados que seriam difíceis de manipular usando os assistentes de serviços da web do CICS.
- Você deseja transferir a manipulação do aplicativo de serviço da web para o zEnterprise Application Assist Processor (zAAP).

Nota: Aplicativos da web implementados em um servidor Liberty JVM não usam os recursos WEBSERVICE ou TCPIPSERVICE. Eles interagem com solicitações da web usando o listener HTTP do Liberty e, portanto, não podem usar os recursos do suporte de serviços da web do CICS.

Validando Mensagens SOAP

Ao usar os serviços da web do CICS, é possível especificar que as mensagens SOAP devem ser validadas para assegurar que estejam em conformidade com o esquema que está contido na descrição do serviço da web. É possível validar tanto aplicativos de modo fornecedor quanto solicitante.

Antes de Iniciar

Durante o desenvolvimento e teste de sua implementação de serviço da web, a validação completa auxilia na detecção de problemas na troca de mensagens entre um solicitante de serviço e um provedor de serviços. No entanto, a validação completa das mensagens SOAP transporta uma sobrecarga substancial e é desaconselhável validar mensagens em um aplicativo de produção totalmente testado.

O CICS usa um programa Java para validar mensagens SOAP. Portanto, deve-se ter o suporte Java ativado em sua região CICS para validar mensagens SOAP.

Sobre Esta Tarefa

A mensagem SOAP é validada antes de ser transformada em uma estrutura de dados do aplicativo e quando uma mensagem SOAP é gerada a partir da estrutura de dados do aplicativo. A mensagem SOAP é validada usando o esquema XML no WSDL e é validada novamente com relação aos requisitos de transformação do CICS. É possível usar o arquivo WSDL especificado no atributo **WSDLFILE** do recurso WEBSERVICE ou um arquivo WSDL contido no arquivo .zip especificado no atributo **ARCHIVEFILE** do recurso WEBSERVICE. Se ambos os atributos forem especificados, o arquivo WSDL no archive especificado no atributo **ARCHIVEFILE** será usado.

Quando a validação está desativada, o CICS não usa o programa Java. O CICS valida mensagens SOAP somente até o limite necessário para confirmar que elas contêm XML bem formado e para transformá-las. Portanto, uma mensagem SOAP pode ser validada com sucesso, mas depois falhar no ambiente de tempo de execução e vice-versa.

Procedimento

1. Configure um servidor JVM OSGi na região CICS. A validação de SOAP usando DFHPIVAL será executada somente em uma estrutura OSGi, não em uma JVM do Axis2 ou de perfil Liberty.
 - a. Instale o servidor JVM de amostra DFH\$JVMS no grupo DFH\$OSGI ou crie seu próprio servidor JVM. Para obter mais informações, consulte Configurando um Servidor JVM.
 - b. Se você criou seu próprio servidor JVM, modifique a definição do programa DFHPIVAL no grupo DFHPIVAL para referenciar o nome do recurso JVMSERVER. A definição DFHPIVAL não é bloqueada e pode ser editada. Por padrão, a definição faz referência ao DFH\$JVMS.
2. Assegure que você tenha uma descrição de serviços da web associado ao recurso WEBSERVICE. Essa associação é criada para recursos WEBSERVICE que são criados automaticamente quando um arquivo WSDL ou um arquivo .zip que contém um ou mais arquivos WSDL está presente no diretório de recebimento do pipeline durante uma varredura de pipeline.

Para definições de WEBSERVICE que são criadas com RDO, a descrição de serviço da web é especificada com o atributo WSDLFILE.
3. Ative a validação de serviço da web especificando o atributo **VALIDATION=YES** do recurso WEBSERVICE. É possível especificar se a validação é necessária quando você define o recurso e é possível mudar essa configuração após o recurso ser instalado.

Resultados

Verifique o log do sistema para descobrir se a mensagem SOAP é válida. A mensagem DFHPI1002 indica que a mensagem SOAP foi validada com sucesso e a mensagem DFHPI1001 indica que a validação falhou.

O que Fazer Depois

Desative a validação quando ela não for mais necessária.

Manipulando dados fornecidos pelo aplicativo inválidos e não inicializados

Como tolerar dados inválidos que são localizados durante a transformação entre JSON e dados do aplicativo.

O serviço que ativa um aplicativo existente resulta no CICS se tornando ciente do conteúdo da COMMAREA ou dos contêineres que são usados por esse aplicativo. O arquivo WSBIND que é implementado no CICS contém informações sobre o formato de dados, incluindo os nomes, o local e o tipo de dados de cada campo individual. O CICS usa essas informações para facilitar a transformação entre a representação de dados XML/JSON e os dados do aplicativo.

Se os dados do aplicativo não forem consistentes com as informações que estão armazenadas no arquivo WSBIND, o CICS relatará um problema e não transformará os dados. Por exemplo, uma interface de aplicativo é mudada, mas o arquivo WSBIND não é regenerado e reimplementado. Esse tipo de problema geralmente resulta em uma mensagem de erro DFHPI1010.

Uma variante mais sutil dessa condição pode ocorrer em cenários em que o aplicativo deixou campos que deliberadamente não são inicializados. Por exemplo, um campo decimal compactado sinalizado pode conter valores baixos (bytes nulos). Quando o CICS processa esse campo, ele não pode determinar que o valor inválido não foi deliberadamente inicializado. O CICS exibe a mensagem de erro DFHPI1010 e o arquivo de transformação de dados. Os aplicativos podem deixar campos não inicializados por muitos motivos. Por exemplo,

- Uma estrutura de linguagem pode descrever os formatos de dados de entrada e saída, mas o campo não inicializado era somente para entrada.
- O aplicativo pode detectar um erro, configurar um código de resposta e retornar sem os outros campos de saída inicializados.
- A lógica condicional no aplicativo pode controlar se o campo é considerado significativo.

A resposta ideal para este cenário é para o aplicativo ser mudado para assegurar que nenhum valor inválido seja transmitido ao CICS para processamento. Uma opção alternativa é configurar a opção DATA-SCREENING do Assistente como DISABLED no momento em que o arquivo WSBIND é gerado. Esta opção faz com que o CICS tolere valores inválidos fornecidos pelo aplicativo e substitua esses valores por um valor padrão, geralmente zero. Essas opções podem tornar a ativação do serviço de aplicativos existentes mais simples, mas tornam a detecção de erro mais difícil e devem ser usadas com cuidado. Se o rastreamento de dados estiver desativado, é mais provável que os erros de dados gerados pelo aplicativo não sejam detectados pelo CICS.

Uma condição relacionada pode ser experimentada onde matrizes geradas pelo aplicativo ficam parcialmente não preenchidas. Por exemplo, considere uma matriz de 1000 registros de dados; o aplicativo pode inicializar os primeiros 10 registros. O CICS gera uma representação de XML ou JSON dos dados e preenche todos os 1000 campos, embora 990 estejam vazios. A resposta ideal para esse problema é introduzir uma cláusula OCCURS DEPENDING ON no aplicativo para indicar precisamente quantos registros foram considerados preenchidos. Uma alternativa é usar a opção TRUNCATE-NUL-ARRAYS dos Assistentes quando o arquivo WSBIND é gerado. Essa opção instrui o CICS a tentar detectar dados não inicializados e a truncar a matriz nesse ponto. O uso dessa opção pode resultar em dados JSON/XML elegantes que são gerados a partir de dados do aplicativo ambíguos e

não inicializados, mas apresenta o risco de perda de dados acidental se o conteúdo da matriz não se distingue de dados não inicializados.

A melhor solução é que os aplicativos produzam dados inequívocos que sejam inteiramente consistentes com a estrutura de linguagem que descreve esses dados. O uso das opções TRUNCATE=NULL-ARRAYS=ENABLED e DATA-SCREENING=DISABLED pode fazer com que o CICS tolere dados fornecidos pelo aplicativo imperfeitos, mas eles introduzem um elemento de risco e incerteza no processo.

O exemplo a seguir demonstra como usar essas opções.

Exemplo 1: tolerância de campos decimais

Testando a correção automática de dados inválidos em um campo decimal.

Sobre Esta Tarefa

Este cenário mostra como o valor padrão 0 pode ser configurado automaticamente em campos decimais que contêm dados inválidos.

Procedimento

1. Gere o esquema JSON e os artefatos necessários para o CICS transformar entre os dados do aplicativo JSON e COBOL.
 - a. Prepare o copybook.

```
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 PBINAR PIC S9(4) BINARY.  
05 NBINAR PIC S9(4) COMP.  
05 UBINAR 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. Submeta o JCL. Configure DATA-SCREENING como DISABLED para ativar a função.


```
//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. Verifique se os esquemas JSON gerados (solicitação e resposta) e arquivos WSBIND foram criados com sucesso.
2. Crie o programa de aplicativo.
 - a. No programa COBOL, designe um valor para todos os campos, exceto decimais.

```
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 do CICS.
 - a. Defina e instale o programa COBOL.
 - b. Defina e instale o TCPIP SERVICE.
 - c. Com base em seu analisador, defina e instale o PIPELINE.
4. Teste o aplicativo.
 - a. Execute PIPELINE SCAN. Para o serviço da web descrito no arquivo WSBIND, use INSERVICE.
 - b. Envie a solicitação JSON.

```
{"BADDATA1operation":{"bad_data":{"normal_num":12}}}
```

- c. Verifique a resposta.

Resultados

Os valores padrão são designados aos campos não inicializados.

```
{
  "BADDATA10perationResponse": {
    "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. Suporte para proteger serviços da web

O CICS Transaction Server para z/OS fornece suporte para várias tecnologias relacionadas que você pode usar para proteger mensagens SOAP e JSON.

Algumas dessas tecnologias estão disponíveis como parte do protocolo HTTP e são igualmente aplicáveis ao SOAP e JSON. Algumas usam a especificação *Web Services Security (WSS): SOAP Message Security 1.0* e estão disponíveis somente para SOAP. Para obter informações sobre as opções de segurança TCP/IP e HTTP compartilhadas, consulte *Segurança para os clientes TCP/IP e Segurança para suporte à Web do CICS*.

Para obter informações sobre como usar asserções SAML, consulte *Visão Geral de Suporte do SAML*.

Segurança de serviços da web SOAP

Web Services Security (WSS): SOAP Message Security 1.0 descreve o uso de *tokens de segurança e assinaturas digitais* para proteger e autenticar mensagens SOAP. Para obter mais informações, consulte a especificação *WSS: Soap Message Security 1.0*.

A Segurança de Serviços da Web protege a *privacidade e integridade* de mensagens SOAP, respectivamente, protegendo mensagens contra divulgação não autorizada e evitando a modificação desautorizada e não detectada. O WSS fornece esta proteção digitalmente assinando e criptografando elementos XML na mensagem. Os elementos que podem ser protegidos são o corpo ou quaisquer elementos no corpo ou no cabeçalho. É possível fornecer diferentes níveis de proteção a diferentes elementos na mensagem SOAP.

A especificação *Web Services Trust Language* aprimora a Segurança de Serviços da Web ainda mais fornecendo uma estrutura para solicitar e emitir tokens de segurança e gerenciar relacionamentos confiáveis entre solicitantes e provedores de serviço da web. Esta extensão para a autenticação de mensagens SOAP permite que serviços da web validem e troquem tokens de segurança de diferentes tipos usando um terceiro confiável. Este terceiro é chamado de *Serviço de Token de Segurança (STS)*. Para obter mais informações sobre a *Web Services Trust Language*, consulte a especificação *WS-Trust Language*.

CICS Transaction Server para z/OS fornece suporte para essas especificações usando um manipulador de segurança fornecido pelo CICS no pipeline:

- Para mensagens não enviadas, o CICS fornece suporte para assinatura digital e criptografia do corpo SOAP inteiro. O CICS também pode trocar um token de nome de usuário por um token de segurança de um tipo diferente com um STS.
- Para mensagens de entrada, o CICS suporta mensagens nas quais o corpo, ou elementos do corpo e cabeçalho, são criptografados ou assinados digitalmente. O CICS também pode trocar e validar tokens de segurança com um STS.

O CICS também fornece uma interface do cliente confiável separada de forma que você possa interagir com um STS sem usar o manipulador de segurança do CICS.

Nota: O *Web Services Security* é potencialmente não compatível com SP800-131A. O *Web Services Security* é configurado, incluindo um manipulador no pipeline e o

CICS não tem controle sobre o processamento em um manipulador gravado pelo cliente. Se você usar assinaturas digitais, poderá especificar somente os algoritmos dsa-sha1 e rsa-sha1. Esses algoritmos não são compatíveis com SP800-131A. O algoritmo de criptografia triplo DES de duas chaves, que pode ser usado para criptografar um corpo SOAP, também não está em conformidade.

Pré-requisitos para Web Services Security

Para implementar Web Services Security, você deve aplicar estas atualizações em sua região do CICS: instale o IBM XML Toolkit for z/OS v1.10, aplique o APAR OA14956 e inclua 3 bibliotecas na concatenação de DFHRPL.

Sobre Esta Tarefa

Conclua as etapas a seguir antes de implementar o Web Services Security:

Procedimento

1. Instale o IBM XML Toolkit for z/OS v1.10 grátis. É possível fazer download dele a partir do site a seguir: <http://www.ibm.com/servers/eserver/zseries/software/xml/>. Você deve instalar a versão 1.10. Versões mais recentes não funcionam com o suporte de Web Services Security no CICS.
2. Aplique o APAR OA14956 do ICSF se ele ainda não estiver instalado no z/OS.
3. Inclua as bibliotecas a seguir na concatenação de DFHRPL:
 - *hlq.SIXMLOD1*, em que *hlq* é o qualificador de alto nível do XML Toolkit.
 - *hlq.SCEERUN*, em que *hlq* é o qualificador de alto nível do Language Environment.
 - *hlq.SDFHWSLD*, onde *hlq* é o qualificador de alto nível da instalação do CICS; por exemplo, CICSTS54.

As primeiras duas bibliotecas contêm DLLs que são necessários no tempo de execução pelo manipulador de segurança. IXM4C57 é fornecido pelo XML Toolkit e está localizado em *hlq.SIXMLOD1*; C128N é fornecido pelo tempo de execução do Language Environment e está localizado em *hlq.SCEERUN*.

A biblioteca *hlq.SDFHWSLD* permite que o CICS localize os módulos DFHWSSE1 e DFHWSXXX do Web Services Security.

4. Pode ser necessário aumentar o valor do parâmetro de inicialização do sistema **EDSALIM**. Os três DLLs que são carregados requerem aproximadamente 15 MB de armazenamento de EDSA.

Resultados

Se você não tiver as bibliotecas especificadas, verá a mensagem a seguir:

CEE3501S O módulo *module_name* não foi localizado.

O *module_name* varia dependendo de qual biblioteca está ausente.

Planejamento para proteger serviços da web SOAP

É possível decidir a melhor maneira de proteger seus serviços da web. O CICS suporta inúmeras opções, incluindo um manipulador de mensagem de segurança configurável e uma interface do cliente de Confiança separada.

Sobre Esta Tarefa

O CICS implementa o Web Services Security (WS-Security ou WSS) em um nível de pipeline, em vez de para cada serviços da web. Responda às perguntas a seguir para decidir a melhor maneira de implementar a segurança.

Procedimento

1. O desempenho de seu processamento de pipeline é importante? O uso do WSS para proteger seus serviços da web incorre em um impacto significativo no desempenho.

A principal vantagem de implementar o WSS é que, criptografando parte de uma mensagem SOAP, é possível enviar a mensagem por meio de uma cadeia de nós intermediários, todos os quais podem ter motivos legítimos para consultar o cabeçalho SOAP para tomar decisões de roteamento ou processamento, mas não têm permissão para visualizar o conteúdo da mensagem. Criptografando somente aquelas seções que precisam ser confidenciais, você deriva os benefícios a seguir:

- Você não incorre na sobrecarga de criptografar e decriptografar em cada nó em uma cadeia de processos intermediários.
- É possível rotear uma mensagem confidencial por uma rede pública de nós não confiáveis, onde somente o destinatário final dos dados pode entendê-la.

Como uma alternativa para usar WSS, é possível usar SSL para criptografar o fluxo de dados inteiro.

2. Se desejar usar WSS, de qual nível de segurança precisa? As opções variam de autenticação básica, na qual o cabeçalho da mensagem inclui um nome de usuário e uma senha, até a combinação de assinaturas digitais e criptografia na mensagem. As opções que o manipulador de segurança do CICS suporta são descritas em “Opções para Proteger Mensagens SOAP”.
3. O manipulador de segurança fornecido pelo CICS atende aos seus requisitos? Se desejar executar o processamento de segurança mais avançado, você deverá gravar seu próprio manipulador de segurança customizado. Este manipulador deve executar a autenticação de mensagens necessária, diretamente com RACF ou usando um Security Token Service, e manipular o processamento de certificados digitais e elementos criptografados. Consulte “Gravando um Manipulador de Segurança Customizada” na página 622 para obter detalhes.
4. Seu pipeline inclui um manipulador MTOM? Se estiver planejando ativar o manipulador MTOM e o manipulador de segurança em seu arquivo de configuração de pipeline, qualquer mensagem MIME Multiparte ou Relacionada será processada no modo de compatibilidade, porque o manipulador de segurança não pode analisar os elementos XOP no corpo da mensagem. Este processamento pode ter um efeito adicional no desempenho do processamento de pipeline.

Opções para Proteger Mensagens SOAP

O CICS suporta a assinatura e a criptografia de mensagens SOAP, portanto, você pode selecionar o nível de segurança que seja mais apropriado para os dados que estão sendo enviados ou recebidos na mensagem SOAP.

A assinatura e criptografia de mensagens SOAP não são suportadas para aplicativos Java de serviço da web Axis2 no modo de provedor ou para serviços da web do provedor que se conectam ao pipeline usando MessageContext do Axis2.

É possível escolher dentre estas opções:

Autenticação Confiável

Em pipelines do provedor de serviços, o CICS pode aceitar um token de nome de usuário no cabeçalho da mensagem SOAP como confiável. Este tipo de token de segurança geralmente contém um nome de usuário e uma senha mas, nesse caso, a senha não é necessária. O CICS confia no nome de usuário fornecido e o coloca no contêiner DFHWS-USERID e a mensagem é processada no pipeline.

Em pipelines do solicitante de serviços, o CICS pode enviar um token de nome do usuário sem a senha no cabeçalho da mensagem SOAP para o provedor de serviços.

Autenticação básica

No modo do provedor de serviços, o CICS pode aceitar um token de nome do usuário no cabeçalho da mensagem SOAP para autenticação em mensagens SOAP de entrada. Este tipo de token de segurança contém um nome de usuário e uma senha. O CICS verifica o token de nome de usuário utilizando um gerenciador de segurança externa, como RACF. Se bem-sucedido, o nome de usuário é colocado no contêiner DFHWS-USERID e a mensagem SOAP é processada no pipeline. Se o CICS não puder verificar o token de nome de usuário, uma mensagem de falha de SOAP será retornada ao solicitante de serviço.

Os tokens de nome de usuário que contêm senhas não são suportados no modo do solicitante de serviço ou em mensagens SOAP de saída.

Autenticação básica HTTP

No modo do provedor de serviços, o CICS pode aceitar informações básicas sobre autenticação por meio de um protocolo HTTP. O solicitante de serviços utiliza uma definição de URIMAP para especificar que as credenciais (informações de identificação do usuário) podem ser capturadas pela saída de usuário global, XWBAUTH. XWBAUTH transmite essas informações para o CICS na solicitação e o CICS envia as informações em um cabeçalho de autorização HTTP para o provedor de serviços.

Autenticação Avançada

Em pipelines do provedor e do solicitante de serviços, é possível verificar ou trocar tokens de segurança por um Security Token Service (STS) para propósitos de autenticação. Essa autenticação permite que o CICS aceite e envie mensagens que têm tokens de segurança no cabeçalho da mensagem que normalmente não são suportados; por exemplo, tokens do Kerberos ou asserções de SAML.

Para uma mensagem de entrada, é possível selecionar para verificar ou trocar um token de segurança. Se a solicitação for para trocar o token de segurança, o CICS deverá receber um token de nome de usuário de volta a partir do STS. Para uma mensagem de saída, é possível trocar um token de nome de usuário somente para um token de segurança.

Assinatura com Certificados X.509

No modo do provedor de serviços e do solicitante de serviço, é possível fornecer um certificado X.509 no cabeçalho da mensagem SOAP para assinar o corpo da mensagem SOAP para autenticação. Este tipo de token de segurança é conhecido como um *token de segurança binário*. Para aceitar tokens de segurança binários a partir de mensagens SOAP de entrada, a chave pública associada ao certificado deve ser importada para um gerenciador de segurança externa, como RACF, e associada ao conjunto de chaves que é especificado no parâmetro de inicialização do sistema

KEYRING. Para mensagens SOAP de saída, você gera e publica a chave pública para os destinatários-alvo. O Integrated Cryptographic Service Facility (ICSF) é usado para gerar chaves públicas.

Ao especificar a etiqueta associada a um certificado digital X.509, não use os caracteres a seguir:

< > : ! =

Também é possível incluir um segundo certificado X.509 no cabeçalho e assiná-lo usando o primeiro certificado. Com este segundo certificado, é possível executar o trabalho no CICS sob o ID do usuário associado ao segundo certificado X.509. O certificado que você está utilizando para assinar a mensagem SOAP deve ser associado a um ID de usuário confiável e ter autoridade substituta para declarar que o trabalho é executado sob uma identidade diferente, a *identidade declarada*, sem o ID do usuário confiável ter a senha associada a essa identidade.

Criptografia

No modo do provedor de serviços e do solicitante de serviço, é possível criptografar o corpo da mensagem SOAP usando um algoritmo simétrico tal como Triple DES ou AES. Um algoritmo simétrico é o local onde a mesma chave é usada para criptografar e decriptografar os dados. Esta chave é conhecida como uma *chave simétrica*. Ela é, então, incluída na mensagem e criptografada utilizando uma combinação da chave pública do destinatário-alvo e o algoritmo de criptografia de chave assimétrica RSA 1.5. Esta criptografia fornece segurança aumentada, porque o algoritmo assimétrico é complexo e é difícil de decriptografar a chave simétrica. No entanto, você obtém melhor desempenho porque a maioria da mensagem SOAP é criptografada com o algoritmo simétrico, que é mais rápido para decriptografar.

Para mensagens SOAP de entrada, é possível criptografar um elemento no corpo SOAP e, em seguida, criptografar o corpo SOAP como um todo. Esse tipo de criptografia poderá ser particularmente apropriado para um elemento que contém dados sensíveis. Se o CICS receber uma mensagem SOAP com dois níveis de criptografia, o CICS decriptografará ambos os níveis automaticamente. Esse tipo de criptografia não é suportado para mensagens SOAP de saída.

O CICS não suporta mensagens SOAP de entrada que possuem um elemento criptografado no cabeçalho da mensagem somente e nenhum elemento criptografado no corpo SOAP.

Assinando e Criptografando

No modo de provedor de serviços e de solicitante de serviço, é possível escolher assinar e criptografar uma mensagem SOAP. O CICS sempre assina o corpo da mensagem SOAP primeiro e, em seguida, criptografa-o. A vantagem deste método é que ele fornece confidencialidade e integridade da mensagem.

Propagação de Identidade Baseada em ICRX

No modo do provedor de serviços, é possível usar um token de identidade ICRX (Extended Identity Context Reference) não autenticado nas mesmas circunstâncias que você usaria um token de ID do usuário do WS-Security não autenticado. Um token de identidade ICRX é um identificador z/OS que mapeia para um ID do usuário. O CICS resolve o token de identidade ICRX para um ID do usuário e coloca uma cópia no contêiner DFHWS-ICRX. O CICS também preenche o contêiner DFHWS-USERID.

Para obter informações adicionais sobre um token de identidade ICRX, consulte Propagação de Identidade e Segurança Distribuída.

Autenticação utilizando um Security Token Service

O CICS pode interoperar com um Security Token Service (STS), tal como o Tivoli Federated Identity Manager, para fornecer autenticação mais avançada de serviços da web.

Um STS é um serviço da web que atua como um terceiro confiável para relacionamentos confiáveis do broker entre um solicitante de serviços da web e um provedor de serviços da web. De forma semelhante, para uma autoridade de certificação em um handshake de SSL, o STS garante que o solicitante e o provedor podem "confiar" nas credenciais fornecidas na mensagem. Essa confiança é representada por meio da troca de tokens de segurança. Um STS pode emitir, trocar e validar estes tokens de segurança e estabelecer relacionamentos confiáveis, permitindo que serviços da web de diferentes domínios de confiança se comuniquem com sucesso. Para obter mais detalhes, consulte a especificação Web Services Trust Language.

O CICS age como um cliente de Confiança e pode enviar dois tipos de solicitação de serviço da web para um STS. O primeiro tipo de pedido é para validar o token de segurança no cabeçalho da mensagem do WS-Security; o segundo tipo é para trocar o token de segurança por um tipo diferente. Estas solicitações permitem que o CICS envie e receba mensagens que contêm diferentes tokens de segurança de uma ampla variedade de domínios de confiança, tais como asserções de SAML e tokens do Kerberos.

É possível configurar o manipulador de segurança do CICS para definir como o CICS interage com um STS ou gravar seu próprio manipulador de segurança para usar uma interface do cliente de Confiança fornecida separadamente. Qualquer que seja o método selecionado, use SSL para proteger a conexão entre o CICS e o STS.

Como o Manipulador de Segurança Chama o STS

O manipulador de segurança do CICS utiliza as informações no arquivo de configuração do pipeline para enviar uma solicitação de serviço da web ao Security Token Service (STS). O tipo de pedido enviado depende da ação que deseja que o STS desempenhe.

Em um enfileiramento do provedor de serviços

Em um pipeline do provedor de serviços, o manipulador de segurança suporta dois tipos de ações, dependendo da maneira como você configura o manipulador de segurança:

- Enviar um pedido ao STS para validar a primeira instância de um token de segurança, ou o primeiro token de segurança de um tipo específico, no cabeçalho do WS-Security da mensagem de entrada.
- Enviar um pedido ao STS para trocar a primeira instância de um token de segurança, ou o primeiro token de segurança de um tipo específico, no cabeçalho do WS-Security da mensagem de entrada, por um token de segurança que o CICS possa compreender.

O manipulador de segurança cria dinamicamente um pipeline para enviar a solicitação de serviço da web ao STS. Esse enfileiramento existirá até que uma resposta seja recebida do STS, após o que será excluída. Se a solicitação for bem-sucedida, o STS retornará um token de identidade ou o

status da validade do token. O manipulador de segurança coloca o RACF ID que é derivado do token no contêiner DFHWS-USERID.

Se o STS encontrar um erro, ele retornará uma falha de SOAP para o manipulador de segurança. O manipulador de segurança, então, transmite uma falha de volta ao solicitante de serviço da web.

Em um enfileiramento do solicitante de serviços

Em um pipeline do solicitante de serviço, o manipulador de segurança pode solicitar somente para trocar um token pelo STS. O arquivo de configuração de pipeline define qual tipo de token o STS emite ao manipulador de segurança.

Se a solicitação for bem-sucedida, o ID do RACF será colocado no contêiner DFHWS-USERID e o token será incluído no cabeçalho da mensagem de saída. Se o STS encontrar um erro, ele retornará uma falha de SOAP para o manipulador de segurança. O manipulador de segurança, então, transmite a falha de volta, por meio do pipeline, ao aplicativo do solicitante de serviço da web.

O manipulador de segurança pode solicitar somente um tipo de ação a partir do STS para o pipeline. Ele também pode trocar somente um tipo de token para uma mensagem de solicitação de saída e está limitado a manipular o primeiro token no cabeçalho da mensagem do WS-Security, a primeira instância ou a primeira instância de um tipo específico. Essas opções abrangem os cenários mais comuns para utilização de um STS, mas pode não oferecer o processamento que requer para manipulação das mensagens de entrada e de saída.

Se desejar fornecer processamento mais específico para manipular muitos tokens nos cabeçalhos das mensagens de entrada ou trocar diversos tipos de tokens para mensagens de saída, utilize a interface do cliente de Confiança. Utilizando essa interface, é possível criar um manipulador de mensagem customizado para enviar sua própria solicitação de serviço da web ao STS.

A Interface do Cliente de Confiança

A interface com o cliente do Trust permite interagir com um STS (Security Token Service) diretamente, em vez de utilizar um manipulador de segurança. Desta maneira, você tem a flexibilidade para fornecer processamento mais avançado de tokens do que o processamento oferecido pelo manipulador de segurança.

A interface com o cliente de confiança é um aprimoramento do DFHPART do programa fornecido pelo CICS. Este programa geralmente é usado para iniciar um pipeline quando um aplicativo do solicitante do serviço da web não tiver sido implementado usando o assistente de serviços da web do CICS. Mas ele também pode agir como a interface do cliente de Confiança para o STS.

É possível chamar a interface com o cliente de confiança ao vincular-se ao DFHPART a partir de um manipulador de mensagem ou programa de processamento de cabeçalho, transmitindo um canal denominado DFHWSTC-V1 e um conjunto de contêineres de segurança. Usando estes contêineres, você tem a flexibilidade de solicitar uma ação de validação ou emissão a partir do STS, selecionar qual tipo de token trocar e transmitir o token apropriado a partir do cabeçalho da mensagem. DFHPART cria dinamicamente um pipeline, compõe uma solicitação de serviço da web a partir de contêineres de segurança e o envia ao STS.

O DFHPIRT espera pela resposta do STS e a transmite de volta no contêiner DFHWS-RESTOKEN para o manipulador de mensagem. Se o STS encontrar um erro, ele retornará uma falha SOAP. O DFHPIRT coloca a falha no contêiner DFHWS-STSFault e o retorna ao programa de ligação no enfileiramento.

É possível utilizar a interface com o cliente do confiança sem ativar o manipulador de segurança no provedor de serviços e nos enfileiramentos do solicitante de serviços ou utilizar a interface com o cliente do confiança como complemento do manipulador de segurança.

Assinatura de Mensagens SOAP

Para mensagens de entrada, o CICS suporta assinaturas digitais em elementos no corpo SOAP e nos blocos de cabeçalho SOAP. Para mensagens de saída, o CICS assina todos os elementos no corpo SOAP.

Uma mensagem SOAP é um documento XML, que consiste em um elemento <Envelope>, que contém um elemento <Header> opcional e um envelope <Body> obrigatório.

A especificação *WSS: SOAP Message Security* permite que o conteúdo do <Header> e do <Body> seja assinado no nível do elemento. Ou seja, em uma mensagem fornecida, elementos individuais podem ser assinados ou não ou podem ser assinados com assinaturas diferentes ou usando algoritmos diferentes. Por exemplo, em uma mensagem SOAP usada em um aplicativo de compra online, é apropriado assinar elementos que confirmam o recebimento de um pedido, porque estes elementos podem ter status válido. No entanto, para evitar a sobrecarga de assinar a mensagem inteira, outras informações podem permanecer com segurança sem assinatura.

Para mensagens de entrada, o manipulador de mensagem de segurança pode verificar a assinatura digital em elementos individuais no <Header> e no <Body> SOAP:

- Elementos assinados que ele encontra no <Header>.
- Elementos assinados no <Body> SOAP. Se o manipulador estiver configurado para esperar um corpo assinado, o CICS rejeitará qualquer mensagem SOAP na qual o corpo não está assinado e emitirá uma falha de SOAP.

Para mensagens de saída, o manipulador de mensagem de segurança pode assinar somente o <Body> SOAP; ele não assinar o <Header>. O algoritmo e a chave usados para assinar o corpo são especificados nas informações de configuração do manipulador.

Algoritmos de Assinatura

O CICS suporta os algoritmos de assinatura requeridos pela especificação XML Signature. Cada algoritmo é identificado por um identificador de recurso universal (URI).

Algoritmo	URI
Algoritmo de Assinatura Digital com Algoritmo Hash Seguro 1 (DSA com SHA1)	http://www.w3.org/2000/09/xmldsig#dsa-sha1
Suportado somente em mensagens SOAP de entrada.	

Suporte CICS para Mensagens SOAP Criptografadas

Para mensagens de entrada, o CICS pode descriptografar quaisquer elementos criptografados no corpo SOAP e blocos de cabeçalho SOAP criptografados nos quais o corpo também é criptografado. Para mensagens de saída, o CICS criptografa o corpo SOAP inteiro.

Uma mensagem SOAP é um documento XML, que consiste em um elemento <Envelope>, que contém um elemento <Header> opcional e um elemento <Body> obrigatório.

A especificação *WSS: SOAP Message Security* permite que alguns dos conteúdos do elemento <Header> e todo o conteúdo do elemento <Body> sejam criptografados no nível do elemento. Ou seja, em uma mensagem fornecida, elementos individuais podem ter diferentes níveis de criptografia ou podem ser criptografados usando diferentes algoritmos. Por exemplo, em uma mensagem SOAP usada em um aplicativo de compra online, é apropriado criptografar os detalhes do cartão de crédito de um indivíduo para assegurar que eles permaneçam confidenciais. No entanto, para evitar a sobrecarga de criptografar a mensagem inteira, algumas informações podem ser criptografadas com segurança usando um algoritmo menos seguro (porém mais rápido) e outras informações podem permanecer descriptografadas com segurança.

Para mensagens de entrada, o manipulador de mensagem de segurança fornecido pelo CICS pode descriptografar elementos individuais no <Body> SOAP e podem descriptografar elementos no <Header> SOAP se o corpo SOAP também é criptografado. O manipulador de mensagem de segurança permite descriptografar estes elementos:

- Elementos que ele encontra no elemento <Header> na ordem na qual os elementos são localizados.
- Elementos no elemento <Body> SOAP. Se desejar rejeitar uma mensagem SOAP que não possui um <Body> criptografado, configure o manipulador para esperar um corpo criptografado usando o elemento <expect_encrypted_body>.

Para mensagens de saída, o manipulador de mensagem de segurança suporta criptografia do conteúdo somente do <Body> SOAP; ele não criptografa nenhum elemento no elemento <Header>. Quando o manipulador de mensagem de segurança criptografa o elemento <Body>, todos os elementos no corpo são criptografados com o mesmo algoritmo e usando a mesma chave. O algoritmo, e informações sobre a chave, são especificados nas informações de configuração sobre o manipulador.

Algoritmos de Criptografia

O CICS suporta os algoritmos de criptografia requeridos pela especificação XML Encryption. Cada algoritmo é identificado por um identificador de recurso universal (URI).

Algoritmo	URI
Algoritmo Triple DES (Data Encryption Standard)	http://www.w3.org/2001/04/xmlenc#tripledes-cbc
Algoritmo AES (Advanced Encryption Standard) com um comprimento de chave de 128 bits	http://www.w3.org/2001/04/xmlenc#aes128-cbc

Configurando o RACF para Web Services Security

Você deve configurar um gerenciador de segurança externa, tal como RACF, para criar pares de chaves pública-privada e certificados X.509 para assinar e criptografar mensagens SOAP de saída e para autenticar e decryptografar mensagens SOAP de entrada assinadas e criptografadas.

Antes de Iniciar

Antes de executar essa tarefa, Deve-se ter o RACF configurado para trabalhar com o CICS. Especifique os parâmetros de inicialização do sistema **DFTUSER**, **KEYRING** e **SEC=YES** na região do CICS que contém seus pipelines de serviço da web.

Nota: Vários certificados com o mesmo Nome Distinto no mesmo **KEYRING** não são suportados.

Procedimento

1. Para autenticar mensagens SOAP de entrada que estão assinadas:
 - a. Importe o certificado X.509 no RACF como uma chave ICSF.
 - b. Conecte o certificado ao conjunto de chaves especificado no parâmetro de inicialização do sistema **KEYRING**, usando o comando **RACDCERT**:

```
RACDCERT ID(userid1)
CONNECT(ID(userid2) LABEL('label-name') RING(ring-name))
```

em que:

- *userid1* é o ID de usuário padrão do conjunto de chaves ou possui autoridade para conectar certificados ao conjunto de chaves para outros IDs do usuário.
 - *userid2* é o ID do usuário que você deseja associar ao certificado.
 - *label-name* é o nome do certificado.
 - *ring-name* é o nome do conjunto de chaves que é especificado no parâmetro de inicialização do sistema **KEYRING**.
- c. Opcional: Se desejar utilizar identidades declaradas, assegure que o ID do usuário associado ao certificado tenha autoridade substituta para permitir que o trabalho seja executado em outros IDs de usuário. Além disso, certifique-se de que quaisquer certificados adicionais incluídos no cabeçalho da mensagem SOAP também sejam importados para o RACF.

A mensagem SOAP pode conter um token de segurança binário no cabeçalho, que inclui o certificado ou contém uma referência ao certificado. Essa referência pode ser o KEYNAME (o rótulo do certificado em RACF), uma combinação do ISSUER e do número SERIAL ou o SubjectKeyIdentifier. O CICS pode reconhecer o SubjectKeyIdentifier somente se ele tiver sido especificado como um atributo na definição do certificado em RACF.

2. Para assinar mensagens SOAP de saída:
 - a. Crie um certificado X.509 e um par de chaves pública-privada utilizando o seguinte comando **RACDCERT**:

```
RACDCERT 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')

```

em que *userid2* é o ID do usuário que deseja associar ao certificado. Quando especificar o valor de *label-name* do certificado, não utilize os seguintes caracteres:

< > : ! =

- b. Conecte o certificado ao conjunto de chaves especificado no parâmetro de inicialização do sistema **KEYRING**. Use o comando **RACDCERT**.
- c. Exporte o certificado e publique-o no destinatário-alvo da mensagem SOAP.

É possível editar o arquivo de configuração de pipeline para que o CICS inclua automaticamente o certificado X.509 no token de segurança binário do cabeçalho da mensagem SOAP para o destinatário-alvo para validar a assinatura.

3. Para decryptografar mensagens SOAP de entrada que são criptografadas, a mensagem SOAP deve incluir a chave pública que faz parte de um par de chaves, no qual a chave privada é definida no CICS.
 - a. Gere um par de chaves pública-privada e um certificado no RACF para criptografia. O par de chaves e o certificado devem ser gerados usando o ICSF.
 - b. Conecte o certificado ao conjunto de chaves especificado no parâmetro de inicialização do sistema **KEYRING**. Use o comando **RACDCERT**.
 - c. Exporte o certificado e publique-o no gerador das mensagens SOAP que você deseja decryptografar.

O gerador da mensagem SOAP pode, então, importar o certificado que contém a chave pública e utilizá-lo para criptografar a mensagem SOAP. A mensagem SOAP pode conter um token de segurança binário no cabeçalho que inclui a chave pública ou contém uma referência a ela. Essa referência pode ser o KEYNAME, uma combinação do ISSUER e do número SERIAL ou o SubjectKeyIdentifier. O CICS pode reconhecer o SubjectKeyIdentifier somente se ele tiver sido especificado como um atributo na definição da chave pública no RACF.

4. Para criptografar mensagens SOAP de saída:
 - a. Importe o certificado que contém a chave pública que você deseja utilizar para criptografia no RACF como uma chave ICSF. O destinatário-alvo deve ter a chave privada associada à chave pública para decryptografar a mensagem SOAP.
 - b. Conecte o certificado que contém a chave pública ao conjunto de chaves especificado no parâmetro de inicialização do sistema **KEYRING**. Use o comando **RACDCERT**.

O CICS usa a chave pública no certificado para criptografar o corpo SOAP e envia o certificado contendo a chave pública como um token de segurança binário no cabeçalho da mensagem SOAP. A chave pública é definida no arquivo de configuração de pipeline.

O que Fazer Depois

Essa configuração para assinar e criptografar mensagens de saída requer que o certificado usado seja de propriedade do ID do usuário da região CICS. O certificado deve pertencer ao ID do usuário da região do CICS porque o RACF

permite que somente o proprietário do certificado extraia a chave privada, a qual é usada para o processo de assinatura e criptografia.

Se o CICS precisa assinar ou criptografar uma mensagem usando um certificado que não possui, é possível compartilhar um único certificado entre sistemas CICS seguindo as instruções em .

Configurando Serviços da Web no Modo de Provedor para Propagação de Identidade

A propagação de identidade com uma solicitação de serviço da web depende de configurações baseadas em confiança; por exemplo, utilizando uma conexão SSL de cliente-certificado a partir do WebSphere DataPower. Nesta tarefa, você configura um recurso PIPELINE para esperar um token de identidade ICRX no cabeçalho do WS-Security, enviado a partir de um cliente confiável.

Antes de Iniciar

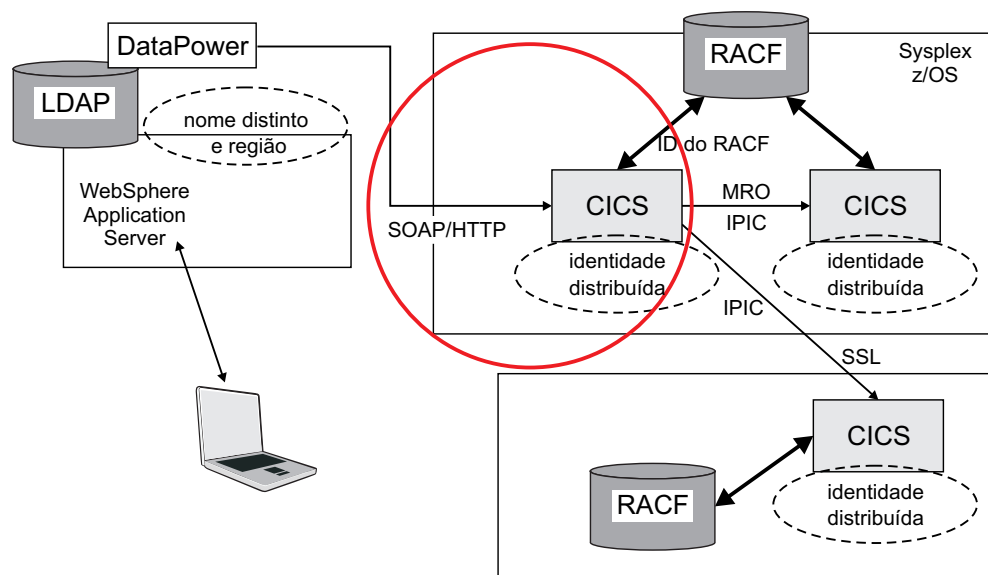
Você deve configurar suas configurações de RACMAP do RACF antes de configurar suas conexões de serviço da web, caso contrário, receberá a mensagem do RACF ICH408I para cada solicitação não mapeada que é enviada ao RACF. Para obter mais informações sobre como configurar o comando **RACMAP** do RACF, consulte Configurando o RACF para Propagação de Identidade.

Você deve configurar um relacionamento *confiável* entre o dispositivo WebSphere DataPower e o CICS, por exemplo, usando certificação do cliente SSL entre WebSphere DataPower e o CICS. O certificado digital que o WebSphere DataPower usa para se identificar deve estar associado a um ID do usuário e esse ID do usuário deve ter autoridade substituta concedida para declarar identidades. Para obter mais informações sobre autoridade substituta, consulte Segurança do Usuário Substituta.

Sobre Esta Tarefa

Esta tarefa explica como usar o CICS com um dispositivo WebSphere DataPower para fornecer uma configuração de serviço da web que pode propagar identidades distribuídas de uma maneira segura e robusta. O círculo no diagrama indica que esta tarefa explica a configuração específica do CICS.

Figura 29. Configurando o CICS para Esperar um Token de Identidade ICRX a Partir do WebSphere DataPower.



WebSphere DataPower age como um intermediário entre o CICS e outros aplicativos. Os aplicativos do solicitante de serviço da web remoto se conectam ao dispositivo WebSphere DataPower usando o protocolo SOAP. O WebSphere DataPower autentica as credenciais fornecidas pelo cliente remoto e mapeia as credenciais para um token de identidade ICRX do z/OS, que identifica a identidade distribuída de um usuário. A mensagem SOAP é, então, encaminhada ao CICS por meio da conexão SSL confiável com um token de identidade ICRX em um cabeçalho WS-Security. Para obter mais informações sobre tokens de identidade ICRX, consulte z/OS Security Server RACF Data Areas.

O CICS recebe a mensagem SOAP do WebSphere DataPower. O arquivo de configuração de PIPELINE especifica a confiança *cega*, porque o único cliente possível é o dispositivo WebSphere DataPower e o WebSphere DataPower está se comunicando com o CICS por meio de uma conexão SSL segura. Portanto, você não precisa especificar autenticação adicional no arquivo de configuração de PIPELINE. O programa manipulador WS-Security localiza o primeiro ICRX localizado no cabeçalho WS-Security e usa o ICRX para identificar o usuário.

Procedimento

1. Crie um recurso PIPELINE, ou edite um recurso PIPELINE existente para especificar o modo ICRX básico, que permite que o PIPELINE receba um ICRX. A combinação mais típica é a confiança cega com o modo ICRX básico. Para obter mais informações sobre o elemento de recurso PIPELINE, consulte O elemento <autenticação>.

A seguir há um arquivo de configuração de PIPELINE de exemplo, mostrando confiança cega com o modo ICRX básico:

```

<?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>
  </service>
</provider_pipeline>

```

```

</service_handler_list>
<terminal_handler>
  <cics_soap_1.2_handler/>
</terminal_handler>
</service>
<apphandler>DFHPITP</apphandler>
</provider_pipeline>

```

A seguir há uma mensagem SOAP de exemplo com uma identidade ICRX, usando confiança cega:

```

<?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. Assegure que o WebSphere DataPower esteja configurado para ser capaz de enviar informações ICRX. Consulte Topologias de rede de amostra para o uso de propagação de identidade.

Resultados

As solicitações de serviço da web do WebSphere DataPower com um token de identidade ICRX no cabeçalho WS-Security, conectado por meio de uma conexão SSL de cliente-certificado, agora podem fluir.

Configurando o Pipeline para Web Services Security

Para configurar um pipeline para suportar Web Services Security (WSS), você deve incluir um manipulador de segurança em seus arquivos de configuração de pipeline. É possível usar o manipulador de segurança fornecido com o CICS, conforme descrito, ou criar seu próprio.

Antes de Iniciar

Antes de definir o manipulador de segurança fornecido pelo CICS, deve-se identificar ou criar os arquivos de configuração de pipeline nos quais você inclui informações de configuração para WSS.

Procedimento

1. Inclua um elemento `<wsse_handler>` para seu pipeline. O manipulador deve ser incluído no elemento `<service_handler_list>` em um pipeline do provedor ou do solicitante de serviços. Codifique os elementos a seguir:

```
<wsse_handler>
  <dfhwsse_configuration version="1">

    </dfhwsse_configuration>
</wsse_handler>
```

O elemento `<dfhwsse_configuration>` é um contêiner para os outros elementos na configuração.

2. Opcional: Codifique um elemento `<authentication>`.
 - Em um pipeline do solicitante de serviço, o elemento `<authentication>` especifica o tipo de autenticação que deve ser usado no cabeçalho de segurança de mensagens SOAP de saída.
 - Em um pipeline do provedor de serviços, o elemento especificar se o CICS usa os tokens de segurança em uma mensagem SOAP de entrada para determinar o ID do usuário sob o qual o trabalho é processado.
 - a. Codifique o atributo **trust** para especificar se a identidade declarada é usada e a natureza do relacionamento confiável entre o provedor e o solicitante de serviços. Para obter detalhes do atributo **trust**, consulte O elemento `<autenticação>`.
 - b. Opcional: Se você especificou **trust=none**, codifique o atributo **mode** para especificar como as credenciais localizadas na mensagem são processadas. Para obter detalhes do atributo **mode**, consulte O elemento `<autenticação>`.
 - c. No elemento `<authentication>`, codifique estes elementos:
 - 1) Um elemento `<suppress/>` vazio, opcional.

Se este elemento for especificado em um pipeline do provedor de serviços, o manipulador não tentará usar nenhum token de segurança na mensagem para determinar sob qual ID do usuário o serviço é executado.

Se este elemento for especificado em um pipeline do solicitante de serviço, o manipulador não tentará incluir na mensagem SOAP de saída qualquer um dos tokens de segurança que são necessários para autenticação.
 - 2) Em um pipeline do solicitante, um elemento `<algorithm>` opcional que especifica o URI do algoritmo que é usado para assinar o corpo da mensagem SOAP. Você deve especificar este elemento se a combinação de valores de atributo de confiança e modo indicar que as mensagens foram assinadas. É possível especificar somente o RSA com o algoritmo SHA1 neste elemento. O URI é <http://www.w3.org/2000/09/xmlsig#rsa-sha1>.
 - 3) Um elemento `<certificate_label>` opcional que especifica a etiqueta que está associada a um certificado digital X.509 instalado no RACF. Se especificar este elemento em um pipeline do solicitante de serviço e o elemento `<suppress>` não for especificado, o certificado será incluído no cabeçalho de segurança na mensagem SOAP. Se você não especificar um elemento `<certificate_label>`, o CICS utilizará o certificado padrão no anel de chaves do RACF.

Este elemento é ignorado em um enfileiramento do provedor de serviços.

3. Opcional: Codifique um elemento `<sts_authentication>` como uma alternativa ao elemento `<authentication>`. Você não deve codificar ambos em seu arquivo de configuração de pipeline. Este elemento especifica que um Security Token Service (STS) é usado para autenticação e determina o tipo de solicitação que é enviada.
 - a. Opcional: Apenas no modo de provedor de serviços, codifique o atributo **action** para especificar se o STS verifica ou troca um token de segurança. Para obter detalhes do atributo **action**, consulte O elemento `<sts_authentication>`.
 - b. Dentro do elemento `<sts_authentication>`, codifique estes elementos:
 - 1) Um elemento `<auth_token_type>`. Este elemento é necessário quando você especifica um elemento `<sts_authentication>` em um pipeline do solicitante de serviço e é ideal em um pipeline do provedor de serviços. Para obter mais informações, consulte `<auth_token_type>`.
 - Em um pipeline do solicitante de serviço, o elemento `<auth_token_type>` indica o tipo de token que o STS emite quando o CICS envia o ID do usuário contido no contêiner DFHWS-USERID. O token que o CICS recebe do STS é colocado no cabeçalho da mensagem de saída.
 - Em um pipeline do provedor de serviços, o elemento `<auth_token_type>` é usado para determinar o token de identidade que o CICS obtém do cabeçalho da mensagem e envia ao STS para trocar ou validar. O CICS usa o primeiro token de identidade do tipo especificado no cabeçalho da mensagem. Se você não especificar este elemento, o CICS usará o primeiro token de identidade que ele localizar no cabeçalho da mensagem. O CICS não considera o seguinte como tokens de identidade:
 - `wsu:Timestamp`
 - `xenc:ReferenceList`
 - `xenc:EncryptedKey`
 - `ds:Signature`
 - 2) Somente em um enfileiramento de provedor de serviços, um elemento `<suppress/>` opcional e vazio. Se esse elemento for especificado, o manipulador não tentará usar nenhum token de segurança na mensagem para determinar o ID do usuário sob o qual o serviço é executado. O elemento `<suppress/>` inclui o token de identidade que é retornado pelo STS.
4. Opcional: Codifique um elemento `<sts_endpoint>`. Use este elemento somente se também tiver especificado um elemento `<sts_authentication>`. No elemento `<sts_endpoint>`, codifique os elementos a seguir:
 - Um elemento `<endpoint>`. Esse elemento contém um URI que aponta para o local do STS (Security Token Service) na rede. É recomendável utilizar o SSL ou o TLS para manter a conexão com o STS segura, em vez de utilizar o HTTP.
Para usar o suporte SAML, configure o terminal como `cics://PROGRAM/DFHSAML`.
Também é possível especificar um terminal do WebSphere MQ usando o formato JMS do URI.
 - Um elemento `<jvmserver>` opcional. Esse elemento identifica o servidor JVM que está configurado para executar o serviço de token SAML. Se esse elemento não for incluído, o servidor JVM do recurso de amostra padrão

DFHXSTS será assumido. Esse elemento é válido somente se você estiver usando SAML: se você usá-lo em outras situações, ocorrerá um erro.

5. Opcional: Se você precisar que as mensagens SOAP de entrada sejam assinadas digitalmente, codifique um elemento `<expect_signed_body/>` vazio. O elemento `<expect_signed_body/>` indica que o `<body>` da mensagem de entrada deve ser assinado. Se o corpo de uma mensagem de entrada não estiver corretamente assinado, o CICS rejeitará a mensagem com uma falha de segurança.
6. Opcional: Se desejar rejeitar mensagens SOAP de entrada que são assinados digitalmente, codifique um elemento `<reject_signature/>` vazio.
7. Opcional: Se precisar que as mensagens SOAP de entrada sejam criptografadas, codifique um elemento `<expect_encrypted_body/>` vazio. O elemento `<expect_encrypted_body/>` indica que o `<body>` da mensagem de entrada deve ser criptografado. Se o corpo de uma mensagem de entrada não estiver corretamente criptografado, o CICS rejeitará a mensagem com uma falha de segurança.
8. Se desejar rejeitar mensagens SOAP de entrada que são criptografadas parcial ou totalmente, codifique um elemento `<reject_encryption/>` vazio.
9. Opcional: Se precisar que mensagens SOAP de saída sejam assinadas, codifique um elemento `<sign_body>`.
 - a. No elemento `<sign_body>`, codifique um elemento `<algorithm>`.
 - b. Após o elemento `<algorithm>`, codifique um elemento `<certificate_label>`.

A seguir há um exemplo de um elemento `<sign_body>` concluído:

```
<sign_body>
  <algorithm>http://www.w3.org/2000/09/xmldsig#rsa-sha1</algorithm>
  <certificate_label>SIGCERT01</certificate_label>
</sign_body>
```

10. Opcional: Se precisar que mensagens SOAP de saída sejam criptografadas, codifique um elemento `<encrypt_body>`.
 - a. No elemento `<encrypt_body>`, codifique um elemento `<algorithm>`.
 - b. Após o elemento `<algorithm>`, codifique um elemento `<certificate_label>`.

A seguir há um exemplo de um elemento `<encrypt_body>` concluído:

```
<encrypt_body>
  <algorithm>http://www.w3.org/2001/04/xmenc#tripledes-cbc</algorithm>
  <certificate_label>ENCCERT02</certificate_label>
</encrypt_body>
```

Exemplo

O exemplo a seguir mostra um manipulador de segurança concluído em que a maioria dos elementos opcionais está presente:

```
<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>
  </dfhwsse_configuration>
</wsse_handler>
```

```

</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>

```

Gravando um Manipulador de Segurança Customizada

Se desejar usar seus próprios procedimentos e processamento de segurança, será possível gravar um manipulador de mensagem customizado para processar mensagens SOAP seguras no pipeline.

Antes de Iniciar

É necessário decidir o nível de segurança que seu manipulador de segurança deve suportar e assegurar que uma falha de SOAP apropriada seja retornada quando uma mensagem incluir segurança que não é suportada.

Sobre Esta Tarefa

O manipulador de mensagem também deve poder lidar com a segurança nas mensagens de entrada e saída.

Procedimento

1. Recupere o contêiner DFHREQUEST ou DFHRESPONSE usando um comando **EXEC CICS GET CONTAINER**.
2. Analise o XML para localizar o token de segurança que está no cabeçalho da mensagem do WS-Security. O cabeçalho inicia com o elemento `<wsse:Security>`. O token de segurança pode ser um nome de usuário e uma senha, um certificado digital ou uma chave de criptografia. Uma mensagem pode ter muitos tokens no cabeçalho de segurança, portanto, seu manipulador precisa identificar o correto para processar.
3. Execute o processamento apropriado, dependendo da segurança que é implementada na mensagem.
 - a. Se você deseja executar autenticação básica de um token Kerberos, emita um comando **EXEC CICS VERIFY TOKEN**. Este comando verifica se o token do Kerberos fornecido é válido. Se o comando for bem-sucedido, atualize o contêiner DFHWS-USERID com um **EXEC CICS PUT CONTAINER**. Caso contrário, emita um comando **EXEC CICS SOAPFAULT CREATE**.
 - b. Se você deseja executar autenticação básica de uma senha ou passphrase, emita um comando **EXEC CICS VERIFY PHRASE**. Este comando verifica o nome de usuário e a senha no cabeçalho de segurança da mensagem. Se o comando for bem-sucedido, atualize o contêiner DFHWS-USERID com um **EXEC CICS PUT CONTAINER**. Caso contrário, emita um comando **EXEC CICS SOAPFAULT CREATE**.
 - c. Se desejar executar a autenticação avançada, trocando ou validando um intervalo de tokens com um Security Token Service, use a interface do cliente de Confiança. Consulte “Chamando o Cliente de Confiança a Partir de um Manipulador de Mensagem” na página 623 para obter detalhes.
 - d. Valide as credenciais do certificado digital se a mensagem for assinada.
 - e. Se partes da mensagem forem criptografadas, decriptografe a mensagem usando as informações no cabeçalho de segurança. A especificação de Como

o CICS está em Conformidade com especificações de Segurança de Serviços da Web fornece informações sobre como fazer isso.

Resultados

Defina seu programa do manipulador de segurança no CICS e atualize o arquivo de configuração de pipeline, assegurando que ele seja colocado corretamente no XML. Em um arquivo de configuração de pipeline do solicitante de serviço, o manipulador de segurança deve ser configurado para executar no final do pipeline. Em um arquivo de configuração do pipeline do provedor de serviços, o manipulador de segurança deve ser configurado para executar no pipeline do pipeline.

O que Fazer Depois

Para obter informações gerais sobre como gravar um manipulador de mensagem personalizada, consulte a publicação IBM Redbooks *Desenvolvimento de aplicativo para CICS Web Services* que está disponível a partir do <http://www.redbooks.ibm.com/abstracts/sg247126.html>.

Chamando o Cliente de Confiança a Partir de um Manipulador de Mensagem

O CICS fornece uma interface para que você possa gravar seu próprio manipulador de mensagem para chamar um Security Token Service (STS). Com esta interface, é possível executar o processamento mais avançado do que o manipulador de segurança fornecido pelo CICS.

Antes de Iniciar

Sobre Esta Tarefa

Você pode utilizar o cliente de Confiança em vez do manipulador de segurança ou além dele. Para usar a interface do cliente de Confiança:

Procedimento

1. Extraia o token correto do cabeçalho da mensagem de segurança da mensagem de entrada ou saída.
2. Vincule-se ao programa DFHPIRT, transmitindo o canal DFHWSTC-V1 e os contêineres necessários a seguir:
 - DFHWS-STSURI, contendo o local do STS na rede.
 - DFHWS-STSACTION, contendo o URI do tipo de solicitação que o STS deve executar. As duas ações suportadas são emitir e validar.
 - DFHWS-IDTOKEN, contendo o token que deve ser verificado ou trocado pelo STS.
 - DFHWS-TOKENTYPE, contendo o tipo de token que o STS deve enviar de volta na resposta.
 - DFHWS-SERVICEURI, contendo o URI da operação de serviço da web que está sendo chamada.

É possível, opcionalmente, incluir o contêiner DFHWS-XMLNS para fornecer os namespaces da mensagem SOAP que contém o token de segurança. Este contêiner é descrito com mais detalhes em A Interface do Programa de Processamento de Cabeçalho.

3. DFHPIRT retorna com a resposta do STS. Uma resposta bem-sucedida é armazenada no contêiner DFHWS-RESTOKEN.
Se o STS encontrar um problema com a solicitação, ele retornará uma falha de SOAP. DFHPIRT coloca a falha de SOAP no contêiner DFHWS-STSFault. Se o STS fornecer um motivo para emitir a falha de SOAP, o motivo será colocado no contêiner DFHWS-STsReason.
Se ocorrer um encerramento de forma anormal, um contêiner DFHError será retornado contendo detalhes do erro de processamento.
Seu manipulador de mensagem deve manipular estas respostas e executar processamento adequado no caso de um erro. Por exemplo, o manipulador de mensagem pode retornar uma falha de SOAP adequada ao solicitante de serviço da web.
4. Processe a resposta conforme apropriado. No modo de provedor, seu processamento de pipeline deve assegurar que um nome de usuário que o CICS possa compreender seja colocado no contêiner DFHWS-USERID quando a mensagem atingir o manipulador de aplicativo. No modo do solicitante, seu manipulador de mensagem deve incluir o token correto no cabeçalho de segurança da mensagem não enviada.

O que Fazer Depois

Quando você tiver gravado seu manipulador de mensagem, implemente o programa no CICS e atualize os arquivos de configuração de pipeline apropriados. Nos pipelines do solicitante de serviço, defina seu manipulador de mensagem para ocorrer no final do processamento de pipeline, mas antes do manipulador de segurança fornecido pelo CICS. Nos pipelines do provedor de serviços, defina seu manipulador de mensagem no início do pipeline, mas após o manipulador de segurança fornecido pelo CICS.

Segurança para o z/OS Connect

O z/OS Connect é um aplicativo WebSphere Liberty e possui a mesma configuração e considerações que outros aplicativos WebSphere Liberty. Além disso, z/OS Connect for CICS 1.0 e z/OS Connect Enterprise Edition têm alguns requisitos de segurança extra.

z/OS Connect for CICS 1.0 e z/OS Connect Enterprise Edition possuem uma interface de gerenciamento RESTful para permitir a descoberta de serviço dinâmico. Esta interface é hospedada no mesmo nome de host e número da porta que os Serviços JSON individuais. O uso de Segurança da Camada de Transporte (TLS) para proteger esta interface e os Serviços JSON individuais são encorajados.

Por padrão, todas as conexões do cliente para z/OS Connect for CICS 1.0 e z/OS Connect Enterprise Edition devem usar o protocolo HTTPS. O comportamento padrão é exigir uma conexão TLS de certificado de cliente com o CICS. Se esse padrão for retido, os certificados de clientes deverão ser associados a um ID de usuário SAF. O aplicativo é executado usando essa identidade derivada de certificado.

O z/OS Connect for CICS 1.0 e o z/OS Connect Enterprise Edition podem ser configurados para suportar o protocolo de autenticação básica HTTP. Esse protocolo permite que um cliente se conecte usando TLS em combinação com um ID do usuário e uma senha SAF. Para ativar o suporte para autenticação básica HTTP, inclua a linha a seguir no arquivo de configuração do servidor Liberty (server.xml): `<webAppSecurity allowFailOverToBasicAuth="true"/>`

Os usuários do z/OS Connect for CICS 1.0 e z/OS Connect Enterprise Edition devem ser um membro do EJBROLE `zos.connect.access.roles.zosConnectAccess`. Para obter informações adicionais, consulte

Consulte Configurando a autorização para aplicativos no Liberty para obter informações do Liberty, Configurando a segurança para o z/OS Connect para obter informações do z/OS Connect e Configurando a segurança para o z/OSÁ Connect EE para obter informações do z/OS Connect EE.

Para obter informações adicionais, consulte e Configurando a Segurança para um Servidor JVM do Liberty.

Configurando permissões para Serviços e APIs do z/OS Connect

O modelo de segurança do CICS requer algumas ações adicionais na maneira como você configura permissões para Serviços e APIs com z/OS Connect for CICS 1.0 e z/OS Connect Enterprise Edition.

Sobre Esta Tarefa

Quando o z/OS Connect é usado para injetar o serviço no CICS, as duas identidades a seguir estão associadas ao serviço em diferentes estágios do processamento:

- Uma identidade provisória inicial é alocada durante o processo de conexão do serviço.
- Uma identidade autenticada é, então, usada para executar o restante do serviço.

É possível configurar essas identidades de várias maneiras, dependendo de suas preferências e do ambiente do sistema.

Procedimento

1. Opcional: Crie um ID do usuário inicial alternativo para o z/OS Connect.

Por padrão, a identidade inicial é o ID do usuário do CICS padrão, mas você pode optar por designar um ID do usuário diferente para evitar dar permissão de ID do usuário do CICS padrão para executar a transação CPIH ou seu equivalente.

- a. Autorize o ID do usuário inicial alternativo para executar a transação CPIH e quaisquer outras transações que são iniciadas por meio do z/OS Connect. O ID do usuário inicial requer permissão para executar a transação de destino para a API ou o Serviço.

2. Designe um ID do usuário inicial padrão. É possível escolher qualquer um ou ambos os métodos a seguir:

- Configure um valor de substituição de ID do usuário no perfil de JVM para o recurso JVMSERVER que hospeda o z/OS Connect.

A seguir está um exemplo de substituição, em que ZOSCUSER é o ID do usuário inicial padrão: `-Dcom.ibm.cics.jvmserver.http.userid=ZOSCUSER`

Nota: Se você configurar um ID do usuário inicial padrão no perfil de JVM, não será necessário fornecer um valor de USERID para cada URIMAP. No entanto, se você fornecer um USERID para um URIMAP e um valor de substituição no perfil de JVM, o USERID especificado para um determinado URIMAP terá precedência.

- Configure o campo USERID para um determinado recurso URIMAP que tem como destino o z/OS Connect.

Quando uma solicitação de HTTP é recebida pelo z/OS Connect, o CICS a corresponde nos recursos URIMAP que estão instalados. Se o URIMAP que é localizado especifica o atributo USERID, esse ID do usuário é usado como o ID do usuário inicial, em vez do ID do usuário inicial padrão para o servidor JVM.

A seguir há um exemplo de configuração para um recurso URIMAP chamado ZOSCDEFT, em que JVMSERVER é o valor de USAGE, um valor genérico é configurado para o atributo PATH, CPIH é a transação de destino e ZOSCUSER é o ID do usuário inicial padrão:

```
NAME: ZOSCDEFT
USAGE: JVMSERVER
SCHEME: HTTP
PORT: NO
HOST: *
PATH: /zosConnect/*
TRANSACTION: CPIH
USERID: ZOSCUSER
```

Nota: Os recursos URIMAP que são instalados usando o mecanismo PIPELINE SCAN provavelmente não são configurados com um ID de usuário padrão. Neste cenário, você pode considerar especificar um valor de substituição de ID do usuário no JVMSERVER.

Nota: É possível armazenar um ID do usuário inicial em um arquivo WSBIND: o usuário de DFHLS2JS ou DFHJS2LS pode fornecer um valor para o parâmetro de entrada **USERID**. Se o parâmetro **USERID** for usado, quaisquer URIMAPs produzidos durante um PIPELINE SCAN incluirão o ID do usuário inicial solicitado.

Resultados

Agora você configurou seu ambiente para que o CICS reconheça os URIs para seus Serviços e APIs e associe um ID do usuário inicial para uso quando a transação de destino estiver conectada.

Capítulo 5. Resolvendo Problemas de Serviços da Web

Os problemas que você pode ter ao implementar serviços da web no CICS podem ocorrer durante o processo de implementação ou no tempo de execução quando o CICS está transformando as mensagens.

Resolução de problemas de serviços da web SOAP

Os problemas que você pode ter ao implementar serviços da web SOAP no CICS podem ocorrer durante o processo de implementação ou no tempo de execução quando o CICS está transformando mensagens SOAP.

Diagnosticando erros de implementação

Os erros de implementação podem ocorrer quando você tenta executar as tarefas em lote do assistente de serviços da web CICS ou as tarefas em lote do assistente XML do CICS, instalar um recurso PIPELINE no CICS ou instalar um recurso WEBSERVICE no CICS. Os erros de implementação mais comuns são descritos aqui, incluindo o sintoma do problema, a causa e a solução.

Sobre Esta Tarefa

No caso de um erro de implementação, recursos PIPELINE geralmente são instalados em um estado DISABLED e recursos WEBSERVICE são instalados em um estado UNUSABLE.

As informações e mensagens de erro associadas às tarefas em lote do assistente de serviço da web do CICS e às tarefas em lote do assistente XML do CICS estão localizadas no log da tarefa. As mensagens de erro associadas à instalação de recursos estão localizadas no log do sistema.

Códigos 0, 4, 8 ou 12 são emitidos pelos assistentes, outros códigos geralmente são emitidos por BPXBATCH, pela JVM ou por IEBGENER.

Os códigos emitidos por BPXBATCH estão em duas categorias principais: um código inferior a 128 indica uma falha de comando, um código superior ou igual a 128 indica que o processo foi finalizado por um sinal. Para obter mais informações sobre BPXBATCH e seus códigos de retorno, consulte a *z/OS UNIX System Services Command Reference*.

Procedimento

- Você recebe um código de retorno 0, 4, 8 ou 12 ao executar as tarefas em lote do assistente de serviços da web do CICS ou as tarefas em lote do assistente XML do CICS. Os códigos de retorno significam o seguinte:
 - 0 - A tarefa foi concluída com sucesso.
 - 4 - Aviso. A tarefa foi concluída com êxito, mas uma ou mais mensagens de aviso foram emitidas.
 - 8 - Erro de entrada. A tarefa não foi concluída com sucesso. Uma ou mais mensagens de erro foram emitidas ao validar os parâmetros de entrada.
 - 12 - Erro. A tarefa não foi concluída com sucesso. Uma ou mais mensagens de erro foram emitidas durante a execução.

1. Verifique o log da tarefa em busca de quaisquer mensagens de aviso ou erro. Consulte as explicações detalhadas para as mensagens. As explicações normalmente descrevem ações que você pode executar para corrigir o problema.
 2. Assegure que você inseriu os valores corretos para cada um dos parâmetros na tarefa. Os valores de parâmetros como nomes de arquivos e elementos na descrição do serviço da web devem ser tratados como distinção entre maiúsculas e minúsculas.
 3. Assegure que você tenha especificado a combinação correta de parâmetros. Por exemplo, se você incluir o parâmetro **PGMNAME** no DFHWS2LS ao gerar um arquivo de ligação de serviço da web para um solicitante de serviços, obterá um erro e a tarefa não será concluída com êxito.
- Você recebe um código de retorno 1, 136 ou 139 ao executar as tarefas em lote do assistente de serviços da web do CICS ou as tarefas em lote do assistente XML do CICS. Esses códigos de retorno indicam que a JVM falhou, geralmente porque há armazenamento insuficiente disponível. Os assistentes do CICS requerem um tamanho da região JCL de pelo menos 300 MB, embora alguns documentos possam requerer 400 MB.
 1. Aumente o tamanho da região ou considere configurar o tamanho da região como 0 M.
 2. Verifique quaisquer saídas de IEFUSI ativas, que podem limitar o tamanho da região.

Nota: Se você usar uma JVM de 64 bits, certifique-se de especificar um valor MEMLIMIT adequado.

- Você recebe um código de retorno 137 ao executar a tarefa em lote do assistente de serviços da web do CICS DFHLS2WS ou a tarefa em lote do assistente XML do CICS DFHLS2SC. Este código de retorno significa que a tarefa atingiu tempo limite.
 1. Aumente o tempo codificando o parâmetro **TIME** na instrução EXEC de sua tarefa como **TIME=1440** ou aumente o valor de MAXCPU TIME no membro SYS1.PARMLIB(BPXPRMxx).
- Você recebe uma mensagem de erro DFHPI0914 ao tentar instalar um recurso WEBSERVICE. A mensagem inclui algumas informações sobre a causa da falha de instalação.
 1. Verifique se você autorizou o CICS a ler o arquivo de ligação do serviço da web no z/OS UNIX.
 2. Verifique se o arquivo de ligação de serviço da web não está corrompido. Isto pode ocorrer, por exemplo, se você usa FTP para transferir o arquivo para o z/OS UNIX no modo de texto em vez de no modo binário.
 3. Verifique se os dois arquivos de ligação de serviço da web com o mesmo nome não estão em diretórios de recebimento diferentes.
 4. Se você estiver tentando instalar um recurso para um aplicativo do solicitante de serviço da web, verifique se a versão da ligação SOAP corresponde ao nível suportado no pipeline. Não é possível instalar um WEBSERVICE SOAP 1.1 em um pipeline do solicitante de serviço que suporta SOAP 1.2.
 5. Verifique se você não está instalando um recurso WEBSERVICE no modo de provedor em um pipeline no modo do solicitante. Arquivos de ligação de serviço da web no modo de provedor especificam um valor de **PROGRAM**, enquanto que arquivos de ligação no modo do solicitante não especificam.

6. Se você estiver usando DFHWS2LS ou DFHLS2WS, verifique se você especificou os parâmetros corretos ao gerar o arquivo de ligação de serviço da web. Alguns parâmetros, tal como **PGMNAME**, são permitidos somente para provedores de serviços da web e devem ser excluídos se você estiver criando um solicitante de serviço da web.
 7. Se você estiver usando DFHWS2LS ou DFHLS2WS, verifique as mensagens emitidas pela tarefa para ver se há algum problema que você precisa resolver antes de criar o recurso WEBSERVICE.
- O recurso PIPELINE falha ao instalar e você recebe um DFHPI0700, DFHPI0712, DFHPI0714 ou uma mensagem de erro semelhante.
 1. Se você recebeu uma mensagem de erro DFHPI0700, é necessário ativar o suporte de linguagem PL/I em sua região CICS. Isto é necessário antes que você possa instalar quaisquer recursos PIPELINE. Consulte Suporte ao ambiente de linguagem para PL/I para obter informações adicionais.
 2. Verifique se você autorizou diretórios do CICS para acessar o z/OS UNIX para ler os arquivos de configuração de pipeline.
 3. Verifique se o diretório que está sendo especificado no parâmetro **WSDIR** é válido. Em específico, verifique se as letras como diretório e nomes do arquivo no z/OS UNIX fazem distinção entre maiúsculas e minúsculas.
 4. Assegure que você não possui um recurso PIPELINE com o mesmo nome em um estado ENABLED na região do CICS.
 - O recurso PIPELINE é instalado em um estado DISABLED. Você recebe uma mensagem de erro no intervalo de DFHPI0702 a DFHPI0711.
 1. Verifique se não há erros no arquivo de configuração de pipeline. Os elementos no arquivo de configuração de pipeline podem aparecer somente em determinados locais. Se você especificá-los incorretamente, receberá uma mensagem de erro DFHPI0702. Esta mensagem inclui o nome do elemento que está causando o problema. Verifique a descrição do elemento para assegurar que você o codificou no local correto.
 2. Verifique se você não possui nenhum caractere não imprimível, tais como guias, no arquivo de configuração de pipeline.
 3. Verifique se o XML é válido. Se o XML não for válido, isto poderá causar erros na análise ao tentar instalar o recurso PIPELINE.
 4. Assegure que o arquivo de configuração de pipeline esteja codificado em US EBCDIC. Se você tentar usar uma codificação EBCDIC diferente, o CICS não poderá processar o arquivo.
 - O recurso WEBSERVICE está em um estado DISABLED. Os estados DISABLED e DISABLING estão disponíveis somente para recursos WEBSERVICE que são definidos e instalados nos pacotes configuráveis do CICS.
 1. Se o recurso PIPELINE associado ao recurso WEBSERVICE tiver sido descartado, o recurso WEBSERVICE entrará no estado DISABLED. Investigue o motivo pelo qual o recurso PIPELINE está ausente e substitua-o, se apropriado.
 2. Se uma ação de desativação tiver sido realizada para o pacote configurável do CICS no qual o recurso WEBSERVICE está definido, o recurso WEBSERVICE entrará no estado DISABLED quando o serviço da web não estiver mais em uso. Investigue o estado do pacote configurável do CICS e ative-o, se apropriado.

Diagnosticando Erros de Tempo de Execução do Provedor de Serviços

Se estiver tendo problemas em receber ou processar mensagens de entrada em um pipeline do modo de provedor, poderá haver um problema com o transporte ou uma mensagem SOAP específica.

Procedimento

- Você receberá uma DFHPI0401, DFHPI0502 ou uma mensagem semelhante, indicando que um erro de transporte HTTP ou do WebSphere MQ ocorreu. Se o transporte for HTTP, o cliente receberá uma mensagem Erro Interno do Servidor 500. Se o transporte for WebSphere MQ, a mensagem será gravada na fila de devoluções (DLQ). Uma falha de SOAP não é retornada ao solicitante de serviços da web, porque o CICS não pode determinar qual tipo de mensagem foi recebido.
- Você receberá uma mensagem DFHxx e uma mensagem de erro 404 Não Localizado.
 1. Se não estiver utilizando o assistente de serviços da web, você deverá criar um recurso URIMAP. Se estiver usando o assistente de serviços da web, o URIMAP será criado automaticamente para você quando executar o comando **PIPELINE SCAN**. O log do sistema fornece informações sobre quaisquer erros que ocorreram como resultado da execução deste comando.
 2. Verifique se o recurso WEBSERVICE está ativado e se o URIMAP ao qual ele está associado é o que você esperava. Se seu recurso WEBSERVICE estiver em um estado UNUSABLE, consulte “Diagnosticando erros de implementação” na página 627.
 3. Verifique se você especificou corretamente o URI e o número da porta. Em específico, verifique as letras, porque o atributo PATH no recurso URIMAP faz distinção entre maiúsculas e minúsculas.
- Se houver erros inesperados sendo relatados, considere a utilização de CEDX para depurar o aplicativo de serviço da web.
 1. Verifique o log do sistema para ver quais mensagens de erro estão sendo relatadas pelo CICS. Isso pode indicar qual tipo de erro está ocorrendo. Se o CICS não estiver relatando nenhum erro, assegure que o pedido esteja atingindo o CICS por meio da rede.
 2. Execute CEDX no CPIH para o transporte HTTP, CPIQ para o transporte do WebSphere MQ ou a transação que você especificou no URIMAP, se for diferente.

Se uma comutação de tarefas ocorrer durante o processamento de pipeline antes do manipulador de aplicativo, a menos que o contêiner DFHWS-TRANID esteja preenchido, a nova tarefa será executada sob o mesmo ID de transação que a primeira. Isso pode interferir na execução de CEDX, porque a primeira tarefa possui um bloqueio na sessão CEDX. É possível evitar este problema utilizando DFHWS-TRANID para alterar o ID da transação quando a tarefa é alternada, permitindo que você utilize CEDX no pipeline e nas tarefas de aplicativos separadamente. Para obter mais informações sobre CEDX, consulte .
 3. Se CEDX não ativar ou permitir que você resolva o problema, considere executar o rastreamento auxiliar com os domínios PI, SO, AP, EI e XS ativos. Isso poderia indicar se existe um problema de segurança, um problema de TCP/IP, um problema de programa de aplicativo ou um problema de pipeline em sua região do CICS. Procure qualquer ponto de rastreamento de exceção ou encerramento anormal.

- Se estiver recebendo erros de conversão, consulte “Diagnosticando Erros de Conversão de Dados” na página 635.
- Se achar que seu problema está relacionado a mensagens MTOM, consulte “Diagnosticando Erros de MTOM/XOP” na página 633.
- Se uma conexão HTTP persistente é encerrada periodicamente:
 1. Verifique se o ajuste de desempenho para conexões HTTP está ativado. Para obter informações adicionais, consulte SOTUNING
 2. Se o ajuste de desempenho for ativado, o CICS fechará periodicamente conexões HTTP persistentes para permitir que as conexões sejam redistribuídas entre regiões que estão atendendo em terminais IP compartilhados.
- Se as tentativas de conexão são recusadas quando a região CICS está na capacidade máxima.
 1. Verifique se o ajuste de desempenho para conexões HTTP está ativado. Para obter informações adicionais, consulte SOTUNING
 2. Se o ajuste de desempenho estiver ativado, quando o CICS estiver na capacidade máxima todas as solicitações abertas de conexão HTTP de entrada serão enfileiradas fora do CICS na fila de lista não processada de conexão de atendimento do TCPIP SERVICE no TCP/IP. O campo Estatísticas globais do TCP/IP SOG_PAUSING_HTTP_LISTENING da região reflete se este é atualmente o caso e os campos SOG_TIMES_AT_ACCEPT_LIMIT/SOG_TIME_LAST_PAUSED_HTTP_LISTENING contêm mais informações sobre ocorrências passadas.
 3. Use **NETSTAT ALL** para obter as estatísticas ConnectionsDropped para a conexão de atendimento do TCPIP SERVICE. Este é o número de solicitações de conexão recebidas e descartadas porque o número de solicitações de conexão em espera na fila de lista não processada atingiu o limite máximo. Para obter mais informações, consulte Netstat nos comandos do administrador do sistema do z/OS Communication Server.
 Estatísticas sobre conexões descartadas também estão disponíveis nos campos a seguir nos serviços TCP/IP: estatísticas do recurso:
 - Conexões Descartadas (SOR_CONNS_DROPPED)
 - Horário em que a conexão foi descartada pela última vez (SOR_CONN_LAST_DROPPED)
 4. Se o valor ConnectionsDropped for muito alto, considere aumentar o valor do atributo de lista não processada do TCPIP SERVICE.

Diagnosticando Erros de Tempo de Execução do Solicitante de Serviço

Leia esta seção se estiver tendo problemas ao enviar solicitações de serviços da web a partir do aplicativo solicitante de serviço ou se estiver recebendo mensagens de falha de SOAP do provedor de serviços da web.

Sobre Esta Tarefa

Problemas que ocorrem podem ser devido a erros em serviços da web individuais ou a problemas no nível de transporte.

Procedimento

- Se você estiver utilizando o comando **INVOKE SERVICE** em seu programa de aplicativo, códigos de RESP e RESP2 serão retornados quando houver um problema.

1. Procure o significado dos códigos RESP e RESP2 para o comando **INVOKE SERVICE** para ter uma indicação do que o problema pode ser.
 2. Verifique o log do sistema CICS para ver se há alguma mensagem que possa ajudá-lo a determinar a causa do problema.
- Se não conseguir enviar uma mensagem de solicitação SOAP e o pipeline estiver retornando um contêiner DFHERROR, houve um problema quando o pipeline tentou processar a mensagem SOAP.
 1. Consulte o conteúdo do contêiner DFHERROR. Ele deve conter uma mensagem de erro e alguns dados que descrevem o problema que ocorreu.
 2. Você introduziu quaisquer novos manipuladores de mensagem ou programas de processamento de cabeçalho no pipeline? Se introduziu, tente remover o novo programa e executar novamente o serviço da web para ver se isso resolve o problema. Se seu manipulador de mensagens estiver tentando executar algum processamento utilizando um contêiner que não está presente no pipeline, ou estiver tentando atualizar um contêiner que é somente leitura, o pipeline parará o processamento e retornará um erro no contêiner DFHERROR. Os programas de processamento de cabeçalho podem atualizar somente um conjunto limitado de contêineres no pipeline. Consulte A Interface do Programa de Processamento de Cabeçalho para obter detalhes.
 3. Se o aplicativo do solicitante de serviço da web não está usando o comando **INVOKE SERVICE** para enviar uma solicitação de serviço da web, verifique se ele criou todos os contêineres de controle necessários e se eles são do tipo de dados correto. Em específico, verifique se o contêiner DFHREQUEST possui um tipo de dados CHAR em vez de BIT.
 4. Se o aplicativo do solicitante de serviço da web está usando o comando **INVOKE SERVICE** e um valor de RESP igual a INVREQ e um código RESP2 igual a 14 é retornado, isso indica que houve um erro de conversão de dados. Consulte “Diagnosticando Erros de Conversão de Dados” na página 635.
 5. Verifique se o XML em sua mensagem SOAP não foi invalidada por um manipulador de mensagem customizado durante o processamento de pipeline. O CICS não executa nenhuma validação em mensagens de saída no pipeline. Se seu aplicativo utilizar o comando **INVOKE SERVICE**, o XML será gerado pelo CICS e estará bem formado quando o corpo da mensagem SOAP for colocado no contêiner DFHREQUEST. No entanto, se você tiver quaisquer manipuladores de mensagem adicionais que alteram o conteúdo da mensagem SOAP, isso não será validado no pipeline.
 - Se você puder enviar uma mensagem SOAP, mas estiver recebendo um erro de tempo limite ou de transporte, isto geralmente é retornado como uma falha de SOAP. Se seu programa estiver usando o comando **INVOKE SERVICE**, o CICS retornará um valor de RESP igual a TIMEDOUT e um código de RESP2 igual a 2 para um erro de tempo limite e um valor de RESP igual a INVREQ e um código de RESP2 igual a 17 para um erro de transporte.
 1. Verifique se o terminal de rede está presente.
 2. Assegure que o atributo RESPWAIT no recurso PIPELINE esteja configurado corretamente para atender aos requisitos de seu aplicativo. O atributo RESPWAIT define por quanto tempo o CICS aguarda por uma resposta do provedor de serviços da web antes de retornar ao aplicativo. Se nenhum valor for especificado, o CICS utilizará os padrões de 10 segundos para HTTP e 60 segundos para o WebSphere MQ. No entanto, o CICS também possui um tempo limite no dispatcher para cada transação e, se ele for menor que o padrão do protocolo que está sendo utilizado, o CICS utilizará o tempo limite do dispatcher no lugar.

- Se puder enviar uma mensagem SOAP, mas estiver recebendo uma resposta de falha de SOAP de volta do provedor de serviços da web que você não esperava, consulte o conteúdo do contêiner DFHWS-BODY para obter detalhes da falha de SOAP.
 1. Se você enviou um envelope SOAP completo em DFHREQUEST utilizando a interface DFHPIRT, assegure que a mensagem de saída não contenha cabeçalhos SOAP duplicados. Isso pode ocorrer quando o pipeline do solicitante utiliza um manipulador de mensagem SOAP 1.1 ou SOAP 1.2. Os manipuladores de mensagem SOAP incluem cabeçalhos SOAP, mesmo se eles já estão especificados no envelope SOAP pelo aplicativo solicitante de serviços. Neste cenário, é possível:
 - Remover o manipulador de mensagem SOAP 1.1 ou SOAP 1.2 do pipeline. Isso afetará quaisquer outros aplicativos do solicitante de serviços que utilizam este pipeline.
 - Remover os cabeçalhos SOAP do envelope SOAP que o aplicativo coloca em DFHREQUEST. O CICS inclui os cabeçalhos SOAP necessários para você. Se deseja executar o processamento adicional nos cabeçalhos, será possível utilizar a interface do programa de processamento de cabeçalho.
 - Usar um comando **WEB SEND** em substituição em seu aplicativo e opte para sair do suporte de serviços da web.
- Se você achar que o problema está relacionado ao envio ou recebimento de mensagens MTOM, consulte “Diagnosticando Erros de MTOM/XOP”.

Diagnosticando Erros de MTOM/XOP

Podem ocorrer erros de MTOM/XOP no tempo de execução, em ambos os pipelines no modo de solicitante e provedor.

Antes de Iniciar

Se você estiver tendo problemas com a configuração de um pipeline para suporte de MTOM/XOP, leia o “Diagnosticando erros de implementação” na página 627.

Sobre Esta Tarefa

Procedimento

- Se você puder enviar uma mensagem de solicitação de serviço da web no formato MTOM, mas estiver recebendo uma mensagem de falha de SOAP do provedor de serviço da web, consulte o conteúdo do contêiner DFHWS-BODY para obter detalhes da falha de SOAP.
 1. O provedor de serviço da web está apto a receber mensagens MIME Multiparte/Relacionadas? Se o provedor de serviço da web não suportar o formato MTOM, a falha que você obtém de volta pode variar dependendo da implementação. Se o provedor de serviços da web for um outro aplicativo CICS, a falha de SOAP poderá indicar que a mensagem MIME não é um tipo de conteúdo válido.
 2. Se o provedor de serviços da web puder receber mensagens MIME, verifique se o pipeline está enviando a mensagem no modo direto ou de compatibilidade. Se você estiver enviando uma mensagem MTOM no modo direto, poderá haver um problema com o XML.
 3. Para descobrir se o problema é com o XML, ative a validação para o serviço da web. Isso faz com que a mensagem do MTOM seja processada no modo de compatibilidade por meio do pipeline. Como parte desse processamento, o manipulador do MTOM analisa o conteúdo da mensagem para otimizar os

dados base64binary. Se houver um erro no XML, o CICS colocará o erro no contêiner DFHERROR e emitirá uma falha de transporte MTOM no pipeline.

4. Examine o conteúdo do contêiner DFHERROR para ver se isso indica qual problema ocorreu. Se não houver informações suficientes para ajudar a diagnosticar a causa do problema, execute um nível de rastreio 2 do domínio PI.
 5. Procure o ponto de rastreio PI 0C16. Isto descreve o problema que foi encontrado com mais detalhes e deve ajudá-lo a corrigir o problema com o XML que é fornecido pelo aplicativo solicitante.
- Se os anexos binários esperados estão ausentes da mensagem MTOM de saída, isso pode indicar que os dados binários são considerados muito pequenos para otimizar como um anexo binário. O CICS cria anexos binários somente para dados que são grandes o suficiente para justificar a sobrecarga do processamento para otimizá-los no pipeline. Quaisquer dados binários abaixo de 1.500 bytes de tamanho não são otimizados.
 - Se você não conseguir enviar uma mensagem MTOM de saída no modo de compatibilidade e o pipeline estiver retornando um contêiner DFHERROR, houve um problema quando o pipeline tentou processar a mensagem MTOM.
 1. Consulte o conteúdo do contêiner DFHERROR. Ele deve conter uma mensagem de erro e alguns dados que descrevem o problema que ocorreu.
 2. Verifique se o XML em sua mensagem MTOM de saída é válida. O CICS não executa nenhuma validação em mensagens de saída no pipeline.
 - Se receber uma mensagem DFHPI1100E, houve um problema com os cabeçalhos MIME de uma mensagem MTOM que foi recebida pelo CICS. A mensagem do CICS contém a classe geral de erro MIME que ocorreu. Para localizar o problema exato que ocorreu:
 1. Se você tiver o rastreio auxiliar ativo em sua região do CICS, verifique se há quaisquer entradas de rastreio de exceção.
 2. Procure o ponto de rastreio PI 1305. Isto descreve a natureza do erro de cabeçalho MIME, o local do erro no cabeçalho e até 80 bytes de texto antes e após o erro para que você possa entender o contexto de onde o erro ocorreu.

Por exemplo, o extrato de rastreio a seguir indica que o parâmetro inicial MIME content-type era inválido porque ele não foi colocado entre aspas, mas incluiu caracteres que não são válidos fora de uma sequência de caracteres entre aspas.

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 68757273 6C65792E 69626D2E 636F6D3E *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 *
```

- O processamento de pipeline falha ao analisar uma mensagem MTOM de entrada e o solicitante de serviços da web recebe uma mensagem de falha de SOAP. Isso indica que houve um problema com o documento XOP na mensagem MTOM. No modo direto, a falha de SOAP é gerada pelo manipulador do aplicativo. Se o pipeline estiver em execução no modo de compatibilidade, a mensagem será analisada pelo manipulador do MTOM ao construir a mensagem SOAP. Neste caso, o CICS emitirá uma mensagem de erro como prefixo DFHPI e uma falha de SOAP.
 1. A mensagem de erro com prefixo DFHPI indica o que estava errado no documento XOP. Por exemplo, poderia ser um cabeçalho MIME inválido ou um anexo binário ausente.

2. Para localizar a causa exata do problema, verifique se há quaisquer pontos de rastreamento de exceção. Em específico, procure por pontos de rastreamento que iniciam com PI 13xx. Isto descreve a exceção que ocorreu em mais detalhes.

Também é possível executar um rastreamento de PI nível 2 para estabelecer a sequência de eventos que levam ao erro, mas isso pode ter um impacto significativo no desempenho e não é recomendado em regiões de produção.

Diagnosticando Erros de Conversão de Dados

Erros de conversão de dados podem ocorrer no tempo de execução ao converter uma mensagem SOAP em uma COMMAREA ou um contêiner do CICS e a partir de uma COMMAREA ou um contêiner em uma mensagem SOAP.

Antes de Iniciar

Os sintomas incluem a geração de mensagens de falha de SOAP e mensagens do CICS indicando que uma falha ocorreu.

Sobre Esta Tarefa

Se você tiver um problema de conversão de dados, execute as etapas a seguir:

Procedimento

1. Assegure que o recurso WEBSERVICE esteja atualizado. Gere novamente o arquivo de ligação de serviço da web para o serviço da web e reimplante-o no CICS.
2. Assegure que o serviço da web remoto tenha sido gerado usando a mesma versão do documento de serviço da web (WSDL) conforme usado ou gerado pelo CICS.
3. Se você tiver certeza que o recurso WEBSERVICE está usando um arquivo de ligação de serviço da web atual:
 - a. Ative a validação de tempo de execução para o recurso WEBSERVICE usando o comando `SET WEBSERVICE(name) VALIDATION` em que *name* é o nome do recurso WEBSERVICE.
 - b. Verifique as mensagens do CICS DFHPI1001 ou DFHPI1002 no log de mensagens. DFHPI1001 descreve a natureza precisa do problema de conversão de dados e pode ajudá-lo a identificar a origem do erro de conversão. DFHPI1002 indica que nenhum problema foi localizado.
 - c. Quando não precisar mais da validação para o serviço da web, use o comando a seguir para desativar a validação: `SET WEBSERVICE(name) NOVALIDATION`.
4. Se você ainda não tiver determinado o motivo para o erro de conversão, execute um rastreamento do CICS do serviço da web com falha. Procure as entradas de rastreamento de exceções do domínio PI a seguir:

```
PI 0F39 - PICC    *EXC* - CONVERSION_ERROR
PI 0F08 - PIII    *EXC* - CONVERSION_ERROR
```

Um erro de conversão de PICC indica que ocorreu um problema ao transformar uma mensagem SOAP de entrada em uma COMMAREA ou um contêiner. Um erro de conversão de PIII indica que ocorreu um problema ao gerar uma mensagem SOAP a partir de uma COMMAREA ou um contêiner fornecido pelo programa de aplicativo. Em ambos os casos, o ponto de rastreamento identifica o nome do campo associado ao erro de conversão e também pode identificar o valor que está causando o problema. Se qualquer um destes

pontos de rastreio ocorrer, eles serão seguidos por um erro de conversão. Para obter uma possível interpretação destes erros de conversão, consulte as explicações das mensagens DFHPI1007 a DFHPI1010.

Por Que Ocorrem Erros de Conversão de Dados

O CICS valida mensagens SOAP somente para a extensão em que é necessário confirmar se elas contêm XML bem formado e para transformá-las. Isto significa que é possível para uma mensagem SOAP ser validada com sucesso usando o WSDL mas, em seguida, falhar no ambiente de tempo de execução e vice-versa.

O recurso WEBSERVICE encapsula as instruções de mapeamento para permitir que o CICS execute a conversão de dados no tempo de execução. Um erro de conversão ocorre quando a entrada não corresponder aos dados esperados, conforme descrito no recurso WEBSERVICE.

Esta incompatibilidade pode ocorrer por qualquer um dos motivos a seguir:

- Uma mensagem SOAP que é recebida pelo CICS não é bem formada e válida quando verificada com relação à descrição de serviços da web (WSDL) associada ao recurso WEBSERVICE.
- Uma mensagem SOAP que é recebida pelo CICS é bem formada e válida, mas contém valores que estão fora do intervalo para o recurso WEBSERVICE.
- O conteúdo de uma COMMAREA ou de um contêiner não é consistente com o recurso WEBSERVICE e a estrutura de linguagem a partir da qual o serviço da web foi gerado.

Por exemplo, o documento WSDL pode especificar restrições de intervalo em um campo, tal como um unsignedInt que pode ter somente um valor entre 10 e 20. Se uma mensagem SOAP contiver um valor igual a 25, a validação da mensagem SOAP fará com que ela seja rejeitada como inválida. O valor 25 é aceito como um valor válido para um número inteiro e é transmitido para o aplicativo.

Um segundo exemplo é onde o documento WSDL especifica uma sequência sem especificar um comprimento máximo. DFHWS2LS assume um comprimento máximo de 255 caracteres por padrão ao gerar o arquivo de ligação de serviço da web. Se a mensagem SOAP contiver 300 caracteres, embora a verificação com relação ao WSDL valide a mensagem como se nenhum comprimento máximo estivesse configurado, um erro será relatado ao tentar transformar a mensagem pois o valor não se ajusta ao buffer de 255 caracteres alocado pelo CICS.

Problemas da Página de Códigos

O CICS usa o valor do parâmetro de inicialização do sistema **LOCALCCSID** para codificar os dados do programa de aplicativo. No entanto, o arquivo de ligação de serviço da web é codificado em US EBCDIC (Cp037). Isto pode levar a problemas com a conversão de dados quando a página de códigos usada pelo programa de aplicativo codifica caracteres de forma diferente para a página de códigos US EBCDIC. Para evitar este problema, é possível usar o parâmetro **CCSID** nas tarefas em lote do assistente de serviços da web para especificar uma página de códigos diferente para codificar dados entre o programa de aplicativo e o arquivo de ligação de serviços da web. O valor deste parâmetro substitui o parâmetro de inicialização do sistema **LOCALCCSID** para esse recurso WEBSERVICE específico. O *value* de **CCSID** deve ser um CCSID EBCDIC.

Mensagens de Falha de SOAP para Erros de Conversão

Se ocorrer um erro de conversão no tempo de execução e o CICS estiver agindo como um provedor de serviço da web, uma mensagem de falha de SOAP será emitida para o solicitante de serviço. Esta mensagem de falha de SOAP inclui a mensagem que é emitida pelo CICS.

O solicitante de serviço pode receber uma das mensagens de falha de SOAP a seguir:

- Não é possível converter mensagens SOAP

Esta mensagem de falha significa que a mensagem SOAP não está bem formada e válida ou seus valores estão fora do intervalo.

- Dados de saída não podem ser convertidos

Esta mensagem de falha significa que o conteúdo de uma COMMAREA ou um contêiner não está consistente.

- Operação não faz parte do serviço da web

Esta mensagem de falha é uma variação especial de quando uma mensagem SOAP inválida é recebida pelo CICS.

Se o CICS for o solicitante de serviço da web, o comando **INVOKE SERVICE** retornará um código de RESP igual a INVREQ e um valor de RESP2 igual a 14.

Resolução de problemas de serviços da web JSON

Os problemas que você pode ter ao implementar serviços da web JSON no CICS podem ocorrer durante o processo de implementação ou no tempo de execução quando o CICS está transformando as mensagens.

Sobre Esta Tarefa

Estas informações de determinação de problema são específicas para o CICS. Se você precisar levantar um problema com o suporte IBM, use as informações nos dados MustGather para assegurar que tenha coletado todas as informações de diagnóstico necessárias.

O suporte para CICS como um provedor de serviços para solicitações JSON é totalmente baseado no suporte do CICS para serviços da web do SOAP.

Resolução de problemas de implementação do JSON

Podem ocorrer erros de implementação quando você tenta instalar um recurso PIPELINE ou WEBSERVICE no CICS. Os erros de implementação mais comuns são descritos aqui, incluindo o sintoma do problema, a causa e a solução.

Procedimento

Os seguintes erros podem ocorrer quando você instala um recurso PIPELINE ou WEBSERVICE no CICS:

- Você recebe uma mensagem de erro DFHPI0914 quando tenta instalar um recurso WEBSERVICE. A mensagem inclui algumas informações sobre a causa da falha de instalação.
 1. Verifique se você autorizou o CICS a ler o arquivo de ligação do serviço da web no z/OS UNIX.

2. Verifique se o arquivo de ligação de serviço da web não está corrompido. Esse dano pode ocorrer, por exemplo, se você usa FTP para transferir o arquivo para z/OS UNIX no modo de texto em vez de no modo binário.
 3. Verifique se os dois arquivos de ligação de serviço da web com o mesmo nome não estão em diretórios de recebimento diferentes.
 4. Verifique se você não está instalando um recurso WEBSERVICE no modo de provedor em um pipeline no modo do solicitante. Arquivos de ligação de serviço da web no modo de provedor especificam um valor de **PROGRAM**, enquanto que arquivos de ligação no modo do solicitante não especificam.
 5. Se você estiver usando DFHJS2LS ou DFHLS2JS, verifique as mensagens que são emitidas pela tarefa para ver se há algum problema que deve ser resolvido antes da criação do recurso WEBSERVICE.
- O recurso PIPELINE falha ao instalar e você recebe DFHPI0700, DFHPI0712, DFHPI0714 ou outra mensagem de erro indicando que o recurso PIPELINE falhou.
 1. Se você recebeu uma mensagem de erro DFHPI0700, deverá ativar o suporte de linguagem PL/I em sua região CICS. Esse suporte é necessário antes que você possa instalar quaisquer recursos PIPELINE. Consulte Suporte ao ambiente de linguagem para PL/I para obter informações adicionais.
 2. Verifique se você autorizou diretórios do CICS para acessar o z/OS UNIX para ler os arquivos de configuração de pipeline.
 3. Verifique se o diretório que está sendo especificado no parâmetro **WSDIR** é válido. Em específico, verifique se as letras como diretório e nomes do arquivo no z/OS UNIX fazem distinção entre maiúsculas e minúsculas.
 4. Assegure que você não possui um recurso PIPELINE com o mesmo nome em um estado ENABLED na região do CICS.
 - O recurso PIPELINE é instalado em um estado DISABLED. Você recebe uma mensagem de erro no intervalo de DFHPI0702 a DFHPI0711.
 1. Verifique se não há erros no arquivo de configuração de pipeline. Os elementos no arquivo de configuração de pipeline podem aparecer somente em determinados locais. Se você especificá-los incorretamente, receberá uma mensagem de erro DFHPI0702. Esta mensagem inclui o nome do elemento que está causando o problema. Verifique a descrição do elemento para assegurar que você o codificou no lugar correto.
 2. Verifique se você não possui nenhum caractere não imprimível, tais como caracteres de tabulação horizontal, no arquivo de configuração de pipeline.
 3. Verifique se o XML é válido. Se o XML não for válido, isto poderá causar erros na análise ao tentar instalar o recurso PIPELINE.
 4. Assegure que o arquivo de configuração de pipeline esteja codificado em US EBCDIC. Se você tentar usar uma codificação EBCDIC diferente, o CICS não poderá processar o arquivo.

Resolução de Problemas do assistente JSON

Se você estiver tendo problemas com o assistente de JSON, utilize as técnicas de resolução de problemas para diagnosticar o problema.

Procedimento

Os seguintes erros podem ocorrer quando você executa o assistente JSON:

- Você recebe um código de retorno 0, 4, 8 ou 12 ao executar as tarefas em lote do assistente CICS JSON. Para obter mais informações sobre os códigos de retorno, consulte “Códigos de retorno do assistente JSON” na página 641.

1. Verifique o log da tarefa em busca de quaisquer mensagens de aviso ou erro. Consulte as explicações detalhadas para as mensagens. As explicações normalmente descrevem ações que você pode executar para corrigir o problema.
 2. Assegure que você inseriu os valores corretos para cada um dos parâmetros na tarefa. Os valores de parâmetros tais como nomes de arquivos e elementos na descrição do serviço da web fazem distinção entre maiúsculas e minúsculas.
 3. Assegure que você tenha especificado a combinação correta de parâmetros.
- Você recebe um código de retorno 1, 136 ou 139 ao executar as tarefas em lote do assistente CICS JSON. Esses códigos de retorno indicam que a JVM falhou, geralmente porque há armazenamento insuficiente disponível. O assistente CICS requer um tamanho da região JCL de pelo menos 300 MB.
 1. Aumente o tamanho da região ou considere configurar o tamanho da região como 0 M.
 2. Verifique quaisquer saídas de IEFUSI ativas, que podem limitar o tamanho da região.
 - Você recebe um código de retorno 137 ao executar a tarefa em lote do assistente CICS JSON. Este código de retorno significa que a tarefa atingiu tempo limite.
 1. Aumente o tempo codificando o parâmetro **TIME** na instrução EXEC de sua tarefa como **TIME=1440** ou aumente o valor **MAXCPU**TIME no membro SYS1.PARMLIB (BPXPRMxx).
 - Você recebe uma mensagem no intervalo de DFHPI9700 a DFHPI9711 indicando um esquema JSON inválido ou não suportado ao executar DFHJS2LS.
 1. Verifique se o esquema JSON é válido. Você pode fazer isso verificando-o com relação à especificação de esquema JSON ou pode usar uma ferramenta, por exemplo json-schema-validator.
 2. Verifique se seu esquema JSON é suportado pelo DFHJS2LS. Para obter mais informações, consulte datamapping.hll-json.cicsassistant.

O que Fazer Depois

Após ter resolvido o problema, execute novamente a tarefa em lote do assistente JSON.

Resolvendo problemas com solicitações do JSON

Se o CICS estiver rejeitando o JSON, seja como uma mensagem de entrada para um serviço da web ou quando usado com a interface vinculável para transformar o JSON, o JSON poderá ser inválido ou incompatível com o esquema.

Sobre Esta Tarefa

Se o CICS encontrar um problema com o JSON fornecido por um solicitante de serviço da web ou um programa de aplicativo, uma mensagem de erro será retornada com detalhes do problema. Quando o CICS está agindo como um provedor de serviços da web, a mensagem de erro é retornada para o aplicativo cliente como uma resposta JSON. Para obter mais informações, consulte “Respostas de erro retornadas ao cliente” na página 640. Quando um aplicativo chama a interface vinculável para transformar o JSON, um código de erro é retornado no contêiner DFHJSON-ERROR e uma mensagem detalhada no contêiner DFHJSON-ERRORMSG. Para obter mais informações, consulte Interface vinculável do transformador JSON. Em ambos os casos as etapas de resolução de problemas são as mesmas e dependem do tipo de erro.

Procedimento

- Verifique se o JSON está em conformidade com as restrições impostas pelo CICS. Para obter mais informações, consulte Restrições de serviço da web JSON.

- Se um erro de análise JSON ocorrer, verifique se o JSON está bem formado. A mensagem de erro fornece detalhes do problema, tal como:

```
"Expected a ',' or '}' at character 44 of {"inquireCatalogRequest":  
"myData } É possível usar uma ferramenta para validar a sintaxe JSON, por  
exemplo JSONLint.
```

- Se o CICS detectar um erro estrutural quando estiver tentando mapear o JSON para dados do aplicativo, uma mensagem DFHPI1007 será emitida, fornecendo detalhes do erro. Por exemplo:

```
DFHPI1007 04/19/2013 15:14:42 IYK2ZKE1 00112 JSON  
to data transformation failed because of incorrect input  
(UNDEFINED_ELEMENT Operation) for WEBSERVICE SimpleMappings.
```

Verifique se o JSON contém os objetos e as propriedades JSON que são descritos pelo esquema fornecido para DFHJS2LS ou que foi gerado pelo DFHLS2JS. Você poderá usar uma ferramenta para validar se o JSON está em conformidade com o esquema.

- Se nenhum erro ocorrer, mas seu aplicativo receber dados vazios para alguns campos, verifique se o JSON correspondente foi fornecido. O CICS não detectará a ausência de propriedades JSON descritas pelo esquema. Você poderia usar uma ferramenta de validação JSON para verificar isso, conforme descrito anteriormente.

- Ocorreu um erro do servidor Interno foi encontrado.

Internal Server Error

CICS TS: Unhandled Pipeline Error

Release: 670

No navegador, verifique a mensagem DFHSJ1006. Verifique se o JVMServer existe e não está desativado.

- Em algumas circunstâncias. O CICS inclui um elemento wrapper na solicitação ou resposta JSON durante o processamento interno. Isso nunca ficará visível para o aplicativo mas, às vezes, poderá ocorrer em mensagens de erro. Por exemplo:

```
"Error obtaining parser from data source:Expected a ':'  
after a key at character 25 of {"DFHWrapper": { 12jb01c":"..."
```

Nessas situações o elemento DFHWrapper deve ser ignorado ao determinar a causa do erro.

O que Fazer Depois

Após ter corrigido o JSON, execute novamente o aplicativo.

Respostas de erro retornadas ao cliente

Essas respostas são retornadas se ocorrerem erros durante o processamento no manipulador CICS JSON.

Sobre Esta Tarefa

Se ocorrerem erros durante o processamento no manipulador CICS JSON, o CICS retornará uma resposta ao cliente contendo informações sobre o erro. Um código

de status HTTP 500 (Erro do Servidor Interno) é retornado e o corpo da mensagem contém detalhes do erro no formato JSON, dependendo do tipo de erro que ocorreu.

Exemplo

Se um erro ocorrer antes ou depois do CICS chamar o pipeline Axis2, a mensagem conterá informações semelhantes ao seguinte:

```
{
  "exception" : {
    "message" : "An exception has occurred while validating HTTP headers".
    "class" : "com.ibm.cicsts.axis2.Controller"
  }
}
```

Se ocorrer um erro em qualquer outro ponto durante o processamento, a mensagem conterá informações semelhantes a uma falha de SOAP, com uma seção de detalhes que varia dependendo da natureza do erro. Isso pode conter uma mensagem do CICS, tal como:

```
{
  "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 do assistente JSON

Se ocorrer uma falha durante a execução das tarefas em lote do assistente JSON, um código de retorno será fornecido indicando o tipo de falha. Essas informações estão contidas no log da tarefa.

Códigos de retorno do programa assistente JSON

No caso de um erro de implementação, recursos PIPELINE geralmente são instalados em um estado DISABLED e recursos WEBSERVICE são instalados em um estado UNUSABLE. Informações e mensagens de erro associadas às tarefas em lote do assistente CICS JSON estão localizadas no log da tarefa. As mensagens de erro associadas à instalação de recursos estão localizadas no log do sistema.

Códigos 0, 4, 8 ou 12 são emitidos pelo assistente JSON, outros códigos geralmente são emitidos por BPXBATCH, pela JVM ou por IEBGENER.

Os códigos emitidos por BPXBATCH estão em duas categorias principais: um código inferior a 128 indica uma falha de comando, um código superior ou igual a 128 indica que o processo foi finalizado por um sinal. Para obter mais informações sobre BPXBATCH e seus códigos de retorno, consulte Referência de Comando do z/OS UNIX System Services.

Você recebe um código de retorno 0, 4, 8 ou 12 ao executar as tarefas em lote do assistente JSON.

Código de Retorno	Descrição
0	A tarefa foi concluída com êxito.
4	Advertência. A tarefa foi concluída com êxito, mas uma ou mais mensagens de aviso foram emitidas.
8	Erro de entrada. A tarefa não foi concluída com sucesso. Uma ou mais mensagens de erro foram emitidas ao validar os parâmetros de entrada.
12	Error. A tarefa não foi concluída com sucesso. Uma ou mais mensagens de erro foram emitidas durante a execução.

Códigos de retorno da tarefa em lote do assistente JSON

Você recebe um código de retorno 1, 136, 137 ou 139 ao executar as tarefas em lote do assistente CICS JSON.

Código de Retorno	Descrição
1	A JVM falhou, geralmente porque há armazenamento insuficiente disponível. O assistente CICS requer um tamanho da região JCL de pelo menos 300 MB.
136	A JVM falhou, geralmente porque há armazenamento insuficiente disponível. O assistente CICS requer um tamanho da região JCL de pelo menos 300 MB.
137	A tarefa expirou.
139	A JVM falhou, geralmente porque há armazenamento insuficiente disponível. O assistente CICS requer um tamanho da região JCL de pelo menos 300 MB.

Apêndice. Amostras de Serviços da Web

O CICS fornece um conjunto de amostras que você pode usar como um ponto de início para o desenvolvimento de aplicativos e a configuração do CICS.

As amostras são categorizadas como a seguir:

O Aplicativo de Exemplo do Gerenciador de Catálogo do CICS

O aplicativo de exemplo do gerenciador de catálogo do CICS é um aplicativo COBOL ativo que foi projetado para ilustrar a melhor prática ao se conectar aos aplicativos CICS para clientes e servidores externos.

O aplicativo base possui uma interface com o usuário 3270, mas a estrutura modular, com interfaces bem definidas entre os componentes, torna possível incluir componentes adicionais. Em específico, o aplicativo é fornecido com suporte de serviço da web, o qual é projetado para ilustrar como é possível estender um aplicativo existente no ambiente de serviços da web.

- “O Aplicativo de Exemplo do Gerenciador de Catálogo do CICS”

Amostras JSON

Utilize estes exemplos para ajudar a entender as solicitações de JSON.

- “Amostras JSON” na página 688

O Aplicativo de Exemplo do Gerenciador de Catálogo do CICS

O aplicativo de exemplo do gerenciador de catálogo do CICS é um aplicativo COBOL ativo que foi projetado para ilustrar a melhor prática ao se conectar aos aplicativos CICS para clientes e servidores externos.

O exemplo é construído em torno de um catálogo de vendas simples e do aplicativo de processamento de pedidos, no qual um usuário pode executar estas tarefas:

- Listar os itens em um catálogo.
- Consultar itens individuais no catálogo.
- Solicitar itens do catálogo.

O catálogo é implementado como um arquivo VSAM.

O aplicativo base possui uma interface com o usuário 3270, mas a estrutura modular, com interfaces bem definidas entre os componentes, torna possível incluir componentes adicionais. Em particular, o aplicativo é fornecido com suporte de serviço da web, o qual é projetado para ilustrar como é possível estender um aplicativo existente no ambiente de serviços da web.

Para este exemplo, o CICS Explorer poderia ser usado para instalar e implementar o aplicativo. O CICS Explorer é uma interface do conjunto de ferramentas baseada em Eclipse para CICS.

Opcional: os usuários que não usam o CICS Explorer podem usar a transação CEDA para corrigir e instalar os recursos contidos nos grupos DFH\$EXBS e DFH\$EXWS.

O Aplicativo Base

O aplicativo base, com sua interface com o usuário 3270, fornece funções com as quais você pode listar o conteúdo de um catálogo armazenado, selecionar um item da lista e inserir uma quantidade a ser pedida. O aplicativo possui um design modular, que torna simples estender o aplicativo para suportar tecnologia mais recente, tais como serviços da web.

Figura 30 mostra a estrutura do aplicativo base.
Os componentes do aplicativo base são:

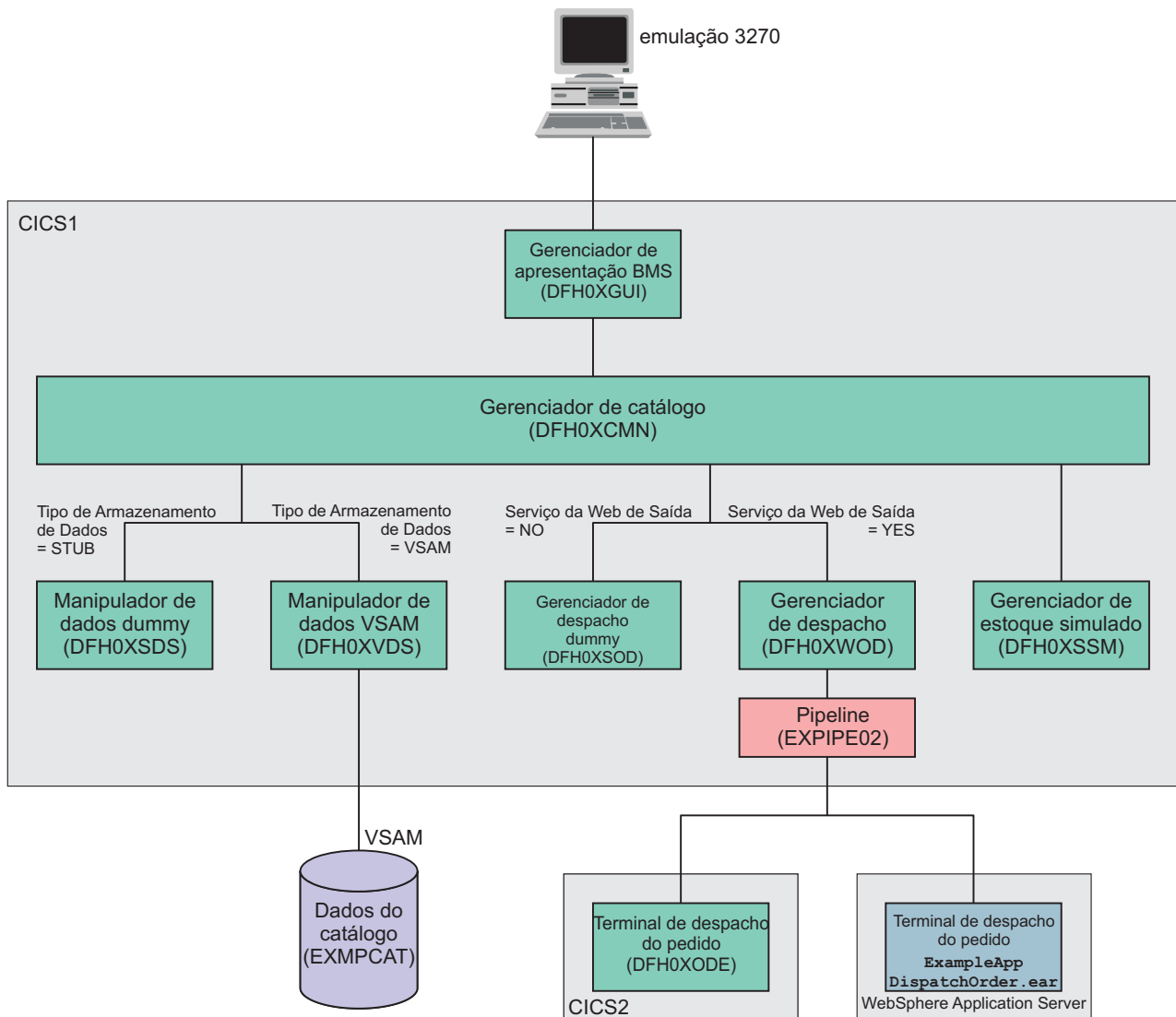


Figura 30. Estrutura do Aplicativo Base

- Um gerenciador de apresentação BMS (DFH0XGUI) que suporta um terminal 3270 ou emulador e que interage com o programa do gerenciador de catálogo principal.
- Um programa do gerenciador de catálogo (DFH0XCMN) que é o núcleo do aplicativo de exemplo e que interage com vários componentes de backend:

- Um programa manipulador de dados que fornece a interface entre o programa do gerenciador de catálogo e o armazenamento de dados. O aplicativo base fornece duas versões deste programa. São elas o programa manipulador de dados VSAM (DFH0XVDS), que armazena dados em um conjunto de dados VSAM; e um manipulador de dados simulado (DFH0XSDS), que não armazena dados, mas retorna respostas válidas ao seu responsável pela chamada. As opções de configuração permitem escolher entre os dois programas.
- Um programa gerenciador de despacho que fornece uma interface para despachar um pedido a um cliente. Novamente, as opções de configuração permitem escolher entre as duas versões deste programa: DFHX0WOD é um solicitante de serviço da web que chama um terminal de despacho de pedido remoto e DFHX0SOD é um programa simulado que retorna respostas válidas ao seu responsável pela chamada.
Há dois terminais de despacho de pedido equivalentes: DFH0XODE é um programa provedor de serviços do CICS; `ExampleAppDispatchOrderV855.war` é um archive web Java que pode ser implementado no servidor JVM do CICS Liberty ou em ambientes semelhantes.
- Um programa gerenciador de estoque simulado (DFH0XSSM) que retorna respostas válidas ao seu responsável pela chamada, mas não executa nenhuma outra ação.

Gerenciador de Apresentação do BMS

O gerenciador de apresentação é responsável por todas as interações com o usuário por meio de painéis BMS 3270. Nenhuma decisão de negócios é tomada neste programa.

O gerenciador de apresentação do BMS pode ser usado de duas maneiras:

- Como parte do aplicativo base.
- Como um cliente de serviço da web do CICS que se comunica com o aplicativo base usando mensagens SOAP.

Manipulador de Dados

O manipulador de dados fornece a interface entre o gerenciador de catálogo e o armazenamento de dados.

O aplicativo de exemplo fornece duas versões da rotina de tratamento de dados:

- A primeira versão usa um arquivo VSAM como o armazenamento de dados.
- A segunda versão é um programa simulado que sempre retorna os mesmos dados em uma consulta e não armazena os resultados de quaisquer solicitações de atualização.

Gerenciador de dispatch

O dispatch de dispatch é responsável por despachar o pedido para o cliente quando o pedido tiver sido confirmado.

O aplicativo de exemplo fornece duas versões do programa do gerenciador de dispatch:

- A primeira versão é um programa simulado que retorna uma resposta correta para o responsável pela chamada, mas não executa nenhuma outra ação.
- A segunda versão é um programa solicitante de serviço da web que faz uma solicitação ao endereço de terminal definido no arquivo de configuração.

Programa de Despacho do Pedido

O programa de despacho do pedido é um programa provedor de serviço da web que é responsável por despachar o item para o cliente.

No aplicativo de exemplo, o dispatcher do pedido é um programa simulado que retorna uma resposta correta ao responsável pela chamada, mas não executa nenhuma outra ação. Isto torna possível para todas as configurações dos serviços da web de exemplo serem operáveis.

Gerenciador de Estoque

O gerenciador de estoque é responsável por gerenciar o reabastecimento do estoque.

No programa de exemplo, o gerenciador de estoque é um programa simulado que retorna uma resposta correta ao responsável pela chamada, mas não executa nenhuma outra ação.

Configuração da Aplicação

O aplicativo de exemplo inclui um programa que permite configurar o aplicativo base.

Instalando e Configurando o Aplicativo Base

Antes de poder executar o aplicativo base, você deve definir e preencher dois conjuntos de dados VSAM e criar dois recursos TRANSACTION.

Criando e Definindo os Conjuntos de Dados VSAM

Os conjuntos de dados KSDS VSAM são usados para definir e preencher o aplicativo de exemplo. Um conjunto de dados contém informações de configuração para o aplicativo de exemplo. O outro contém o catálogo de vendas.

Procedimento

1. Localize a JCL para criar os conjuntos de dados VSAM. Durante a instalação do CICS, a JCL é colocada na biblioteca *hlq.SDFHINST*:
 - O membro DFH\$ECNF contém a JCL para gerar o conjunto de dados de configuração.
 - O membro DFH\$ECAT contém a JCL para gerar o conjunto de dados de catálogo.
2. Modifique a JCL e os comandos de serviços do método de acesso.
 - a. Forneça uma placa JOB válida.
 - b. Forneça um qualificador de alto nível adequado para os nomes do conjunto de dados nos comandos de Access Method Services. Conforme fornecida, a JCL usa um qualificador de alto nível de HLQ.

O comando a seguir define o arquivo de configuração:

```
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) -  
    )
```

em que *hlq* é um qualificador de alto nível de sua escolha.

O comando a seguir define o arquivo do catálogo:

```
DEFINE CLUSTER (NAME(hlq.EXAMPLAPP.catname)-  
  TRK(1 1) -  
  KEYS(4 0) -  
  RECORDSIZE(80,80) -  
  SHAREOPTIONS(2 3) -  
  INDEXED -  
) -  
DATA (NAME(hlq.EXAMPLAPP.catname.DATA) -  
) -  
INDEX (NAME(hlq.EXAMPLAPP.catname.INDEX) -  
)
```

em que:

- *hlq* é um qualificador de alto nível de sua escolha
- *catname* é um nome de sua escolha. O nome utilizado no aplicativo de exemplo conforme fornecido é EXMPCAT.

3. Execute ambas as tarefas para criar e preencher os conjuntos de dados.
4. Use o CICS Explorer para criar uma definição de FILE para o arquivo de catálogo.
 - a. Selecione **Definições > Definições de Arquivo**. Clique com o botão direito na coluna **Nome** e clique em **Novo** para criar uma nova definição de arquivo. Digite EXEMPLO na caixa de texto **Grupo de Recursos** e digite EXMPCAT na caixa de texto **Nome**. Clique em **Concluir** para definir a definição de FILE. Como alternativa, é possível copiar a definição de arquivo do grupo fornecido pelo CICS, DFH\$EXBS.
 - b. Dê um clique duplo no novo arquivo EXMPCAT. No editor Aplicativo de Exemplo CICS de Definição de Arquivo (EXMPCAT), selecione a guia **VSAM**. Digite *hlq*.EXAMPLAPP.EXMPCAT na caixa de texto **Nome do Conjunto de Dados a ser Usado**.
 - c. Selecione a guia **Atributos** e configure as operações dos atributos a seguir como **Sim**:
 - Adicionar
 - Pesquisar
 - Delete
 - Leitura
 - Atualizar
 - d. Use os valores padrão para todos os outros atributos.
5. Use o CICS Explorer para criar uma definição de FILE para o arquivo de configuração.
 - a. Selecione **Definições > Definições de Arquivo**. Clique com o botão direito na coluna **Nome** e clique em **Novo** para criar uma nova definição de arquivo. Digite EXEMPLO na caixa de texto **Grupo de Recursos** e digite EXMPCONF na caixa de texto **Nome**. Clique em **Concluir** para definir a definição de FILE. Como alternativa, é possível copiar a definição de arquivo do grupo fornecido pelo CICS, DFH\$EXBS.
 - b. Dê um clique duplo no novo arquivo EXMPCONF. Na janela Aplicativo de Exemplo CICS de Definição de Arquivo (EXMPCONF), selecione a guia **VSAM**. Digite *hlq*.EXAMPLAPP.EXMPCONF na caixa de texto **Nome do Conjunto de Dados a ser Usado**.
 - c. Selecione a guia **Atributos** e configure as operações dos atributos a seguir como **Sim**:
 - Adicionar

Pesquisar
Delete
Leitura
Atualizar

- d. Use os valores padrão para todos os outros atributos.

Resultados

Os conjuntos de dados foram preenchidos e as definições de FILE para o arquivo de catálogo e o arquivo de configuração foram criadas e estão prontas para instalação.

Opcional: os usuários que não usam o CICS Explorer podem usar a transação CEDA para corrigir e instalar os recursos contidos nos grupos DFH\$EXBS e DFH\$EXWS.

Definindo a Interface 3270

O aplicativo de exemplo é fornecido com uma interface com o usuário 3270 para executar o aplicativo e customizá-lo. A interface com o usuário consiste em duas transações, EGUI e ECFG. Uma terceira transação, ECLI, é usada para o cliente de serviço da web do CICS.

Sobre Esta Tarefa

Opcional: os usuários que não usam o CICS Explorer podem usar a transação CEDA para corrigir e instalar os recursos contidos nos grupos DFH\$EXBS e DFH\$EXWS.

Procedimento

1. Crie definições de TRANSACTION para as transações a seguir utilizando o CICS Explorer. A operação correta do aplicativo de exemplo não depende dos nomes das transações, portanto, você pode utilizar nomes diferentes.
 - a. Copie as definições para a transação EGUI a partir do grupo DFH\$EXBS fornecido pelo CICS clicando com o botão direito do mouse na visualização Definições do Grupo de Recursos. Selecione **Novo > Definição de Transação**. Digite EGUI na caixa de texto **Nome** e DFH0XGUI na caixa de texto **Nome do Programa**. Clique em **Concluir** para criar a definição de EGUI TRANSACTION.
 - b. Copie as definições para a transação ECFG a partir do grupo DFH\$EXBS fornecido pelo CICS clicando com o botão direito do mouse em DFH\$EXBS na janela Definições do Grupo de Recursos. Selecione **Novo > Definição de Transação**. Digite ECFG na caixa de texto **Nome** e DFH0XCFG na caixa de texto **Nome do Programa**. Clique em **Concluir** para criar a definição de ECFG TRANSACTION.
 - c. Opcional: Copie as definições para transações ECLI a partir do grupo DFH\$EXWS fornecido pelo CICS clicando com o botão direito do mouse em DFH\$EXWS na visualização Definições do Grupo de Recursos. Selecione **Novo > Definição de Transação**. Digite ECLI na caixa de texto **Nome**, e DFH0XCUI na caixa de texto **Nome do Programa**. Clique em **Concluir** para criar a definição de transação ECLI.

Use os valores padrão para todos os outros atributos.

2. Opcional: Se você não desejar usar a instalação automática do programa, copie as definições de PROGRAM para os programas de aplicativo base e as definições de MAPSET para os mapas BMS a partir do grupo DFH\$EXBS fornecido pelo CICS.
 - a. Copie as definições de recurso MAPSET para os mapas BMS nos membros DFH0XS1, DFH0XS2 e DFH0XS3. Para obter detalhes do que está em cada membro, consulte “Componentes do Aplicativo Base” na página 678.
 - b. Copie as definições de recurso de PROGRAM para os programas COBOL a seguir.

Tabela 16. Membros SDFHSAMP Contendo Origem COBOL para o Aplicativo Base

Member name	Descrição
DFH0XCFG	Programa chamado pela transação ECFG para ler e atualizar o arquivo de configuração do VSAM.
DFH0XCMN	Programa controlador para o aplicativo de catálogo. Todas as solicitações passam pelo programa controlador.
DFH0XGUI	Programa chamado pela transação EGUI para gerenciar o envio dos mapas BMS para o usuário do terminal e o recebimento dos mapas do usuário do terminal. Este programa se vincula ao programa DFH0XCMN.
DFH0XODE	Uma das duas versões do terminal para o serviço da web de envio do pedido. Esta é a versão que é executada no CICS. Este programa configura o texto "Order in dispatch" na COMMAREA de retorno.
DFH0XSDS	Uma versão <i>em stub</i> ou simulada do programa de armazenamento de dados que permite ao aplicativo trabalhar quando o arquivo de catálogo do VSAM não tiver sido configurado. DFH0XSDS usa dados definidos no programa em vez de dados armazenados em um arquivo VSAM.
DFH0XSOD	Uma versão em stub do programa de envio do pedido. Ele configura o código de retorno na COMMAREA como 0 e retorna ao seu responsável pela chamada. DFH0XSOD é usado quando serviços da web de saída não são necessários.
DFH0XSSM	Uma versão em stub do programa gerenciador de estoque (reabastecimento). DFH0XSSM configura o código de retorno na COMMAREA como 0 e retorna ao seu responsável pela chamada.
DFH0XVDS	A versão de VSAM do programa de armazenamento de dados. DFH0XVDS acessa o arquivo VSAM para executar leituras e atualizações do catálogo.
DFH0XWOD	A versão de serviço da web do programa de despacho do pedido. DFH0XWOD emite um EXEC CICS INVOKE WEBSERVICE para fazer uma chamada de serviço da web de saída para um dispatcher de pedido.

Use os valores padrão para todos os outros atributos.

- c. Opcional: Copie as definições de PROGRAM para DFH0XCUI a partir do grupo DFH\$EXWS fornecido pelo CICS. Use os valores padrão para todos os outros atributos. Este programa é requerido se você deseja utilizar a transação ECLI que inicia o cliente de serviço da web.

```
DIS G(DFH$EXWS)
ENTER COMMANDS
NAME      TYPE      GROUP
DFH0XCUI PROGRAM    DFH$EXWS
```

ECLI	TRANSACTION	DFH\$EXWS
EXMPPORT	TCPIP SERVICE	DFH\$EXWS
EXPIPE01	PIPELINE	DFH\$EXWS
EXPIPE02	PIPELINE	DFH\$EXWS

Concluindo a Instalação

Para concluir a instalação, instale o grupo RDO que contém suas definições de recurso.

Procedimento

Clique com o botão direito do mouse no grupo de recursos na janela Definições do Grupo de Recursos. Selecione **Instalar**. Certifique-se que seu CICSplex esteja correto e que você selecionou sua região de destino, em seguida, clique em **OK**.

Resultados

Seu RDO agora está instalado e o aplicativo está pronto para uso.

Opcional: os usuários que não usam o CICS Explorer podem usar a transação CEDA para corrigir e instalar os recursos contidos nos grupos DFH\$EXBS e DFH\$EXWS.

Configurando o Aplicativo de Exemplo

O aplicativo base inclui uma transação (ECFG) que você pode utilizar para configurar o aplicativo de exemplo.

Antes de Iniciar

A transação de configuração usa informações compostas por letras maiúsculas e minúsculas. Você deve usar um terminal que pode manipular informações compostas por letras maiúsculas e minúsculas corretamente.

Sobre Esta Tarefa

É possível especificar vários aspectos do aplicativo de exemplo. Elas incluem:

- A configuração geral do aplicativo, tal como o uso de serviços da web
- Os endereços de rede usados pelos componentes de serviços da web do aplicativo
- Os nomes de recursos, tal como o arquivo usado para o armazenamento de dados
- Os nomes de programas usados para cada componente do aplicativo

Com a transação de configuração, é possível substituir componentes fornecidos pelo CICS do aplicativo de exemplo por seu próprio sem reiniciar o aplicativo.

Procedimento

1. Digite a transação ECFG para iniciar o aplicativo de configuração. O CICS exibe a seguinte tela:

CONFIGURE CICS EXAMPLE CATALOG APPLICATION

```
Datastore Type ==> VSAM          STUB|VSAM
Outbound WebService? ==> NO      YES|NO
Catalog Manager ==> DFH0XCMN
Data Store Stub ==> DFH0XSDS
Data Store VSAM ==> DFH0XVDS
Order Dispatch Stub ==> DFH0XSOD
Order Dispatch WebService ==> DFH0XWOD
Stock Manager ==> DFH0XSSM
VSAM File Name ==> EXMPCAT
Server Address and Port ==> myserver:99999
Outbound WebService URI ==> http://myserver:80/exampleApp/dispatchOrder
                        ==>
                        ==>
                        ==>
                        ==>
```

PF

3 END

12 CNCL

2. Conclua esses campos.

Tipo de armazenamento de dados

Especifique STUB se desejar usar o programa stub de armazenamento de dados.

Especifique VSAM se desejar usar o programa de armazenamento de dados VSAM.

Outbound WebService

Especifique YES se desejar usar um serviço da web remoto para sua função de despacho de pedido; ou seja, se deseja que o programa do gerenciador de catálogo se vincule ao programa de serviço da web de despacho do pedido.

Especifique NO se desejar usar um programa stub para sua função de despacho de pedido; ou seja, se deseja que o programa do gerenciador de catálogo se vincule ao programa stub do despacho de pedido.

Catalog Manager

Especifique o nome do programa do gerenciador de catálogo. O programa fornecido com o aplicativo de exemplo é DFH0XCMN.

Data Store Stub

Se você especificou STUB no campo **Datastore Type**, especifique o nome do programa stub de armazenamento de dados. O programa fornecido com o aplicativo de exemplo é DFH0XSDS.

Data Store VSAM

Se você especificou VSAM no campo **Datastore Type**, especifique o nome do programa de armazenamento de dados VSAM. O programa fornecido com o aplicativo de exemplo é DFH0XVDS.

Order Dispatch Stub

Se você especificou NO no campo **Outbound WebService**, especifique o nome do programa stub de despacho do pedido. O programa fornecido com o aplicativo de exemplo é DFH0XSOD.

Order Dispatch WebService

Se você especificou YES no campo **Outbound WebService**, especifique o

nome do programa que funciona como um solicitante de serviço. O programa fornecido com o aplicativo de exemplo é DFH0XWOD.

Stock Manager

Especifique o nome do programa gerenciador de estoque. O programa fornecido com o aplicativo de exemplo é DFH0XSSM.

VSAM File Name

Se você especificou VSAM no campo **Datastore Type**, especifique o nome da definição de CICS FILE. O nome utilizado no aplicativo de exemplo conforme fornecido é EXMPCAT.

Server Address and Port

Se você estiver usando o cliente de serviço da web do CICS, especifique o endereço IP e a porta do sistema no qual o aplicativo de exemplo é implementado como um serviço da web.

Outbound WebService URI

Se você especificou YES no campo **Outbound WebService**, especifique o local do serviço da web que implementa a função do pedido de despacho. Se estiver usando o terminal CICS fornecido, configure o **Outbound WebService** como: `http://myserver:myport/exampleApp/dispatchOrder` em que *myserver* e *myport* são seu endereço do servidor e sua porta do CICS.

Executando o Aplicativo de Exemplo com a Interface BMS

O aplicativo base pode ser executado usando a interface BMS.

Procedimento

1. Digite a transação EGUI a partir de um terminal CICS. O menu do aplicativo de exemplo é exibido:

```
CICS EXAMPLE CATALOG APPLICATION - Main Menu
```

```
Select an action, then press ENTER
```

```
Action . . . . 1. List Items
                2. Order Item Number ____
                3. Exit
```

```
F3=EXIT    F12=CANCEL
```

É possível listar os itens no catálogo, solicitar um item ou sair do aplicativo usando as opções no menu.

2. Digite 1 e pressione Enter para selecionar a opção List Items. O aplicativo exibe uma lista de itens no catálogo.

CICS EXAMPLE CATALOG APPLICATION - Inquire Catalog

Select a single item to order with /, then press ENTER

Item	Description	Cost	Order
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 201b 500/ream	5.35	-
0100	Green Laser Paper 201b 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=EXIT F7=BACK F8=FORWARD F12=CANCEL

3. Digite / na coluna **Order** e pressione Enter para solicitar um item. O aplicativo exibe detalhes do item a ser solicitado.

CICS EXAMPLE CATALOG APPLICATION - Details of your order

Enter order details, then press ENTER

Item	Description	Cost	Stock	On Order
0010	Ball Pens Black 24pk	2.90	0011	000

Order Quantity: 5
User Name: CHRISB
Charge Dept: CICSDEV1

F3=EXIT F12=CANCEL

4. Se houver estoque suficiente para atender ao pedido, insira as informações a seguir:
 - a. Preencha o campo **Order Quantity**. Especifique o número de itens que deseja solicitar.
 - b. Preencha o campo **User Name**. Insira uma sequência de 1 a 8 caracteres. O aplicativo de base não verifica o valor digitado aqui.
 - c. Preencha o campo **Charge Dept**. Insira uma sequência de 1 a 8 caracteres. O aplicativo de base não verifica o valor digitado aqui.
5. Pressione Enter para enviar o pedido e retornar ao menu principal.
6. Pressione F3 para finalizar os aplicativos.

Suporte de Serviço da Web para o Aplicativo de Exemplo

O suporte de serviço da web estende o aplicativo de exemplo, fornecendo duas versões Java de um cliente de front-end do servidor da web e uma versão Java e COBOL do terminal da web em serviço para o componente do dispatcher de pedido.

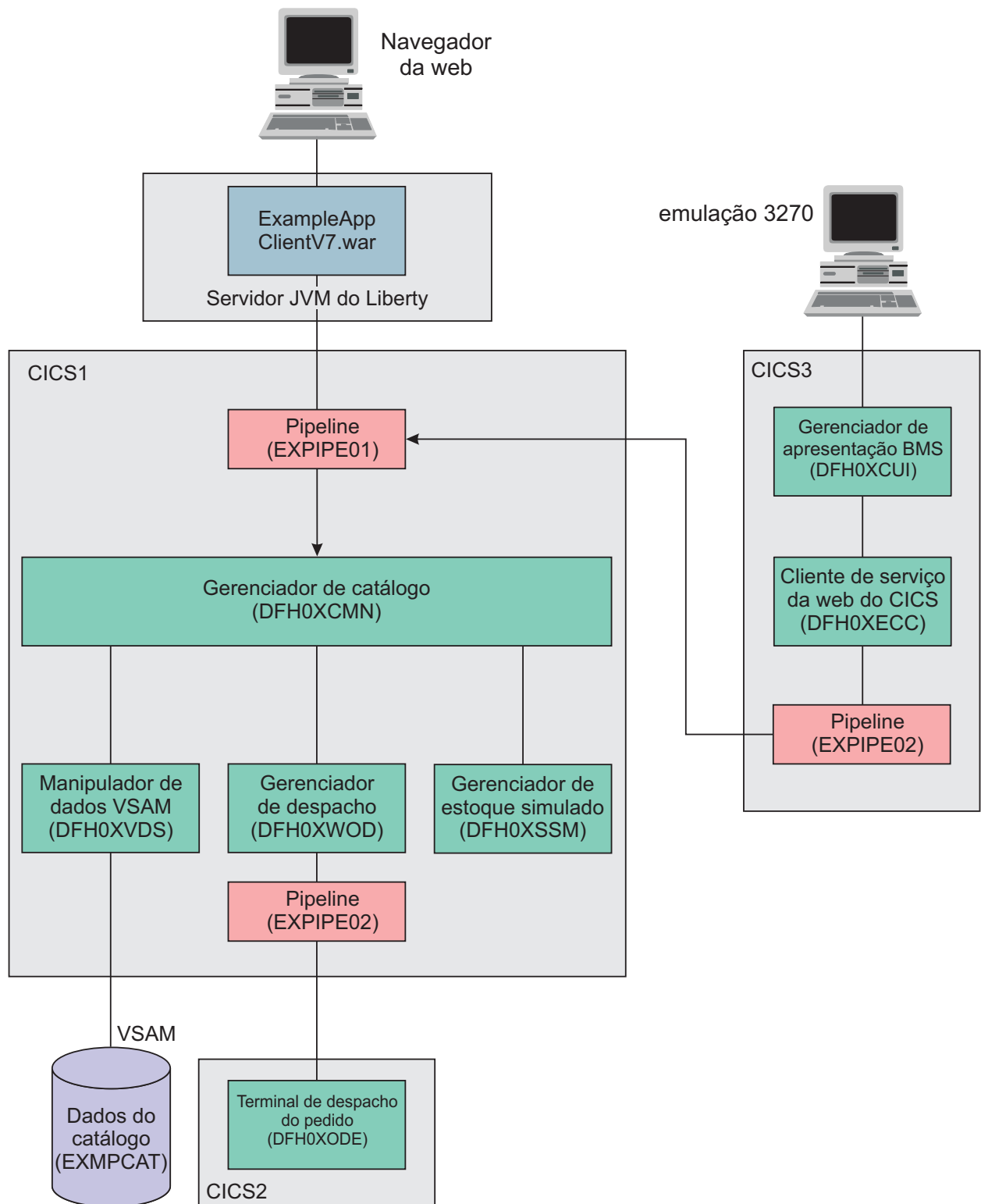
As duas versões do front-end do cliente da web e uma versão do terminal da web em serviço são fornecidas como arquivos Java web archive (WARs) que são executados no ambiente de perfil da web Java EE 6 fornecido pela versão mais recente do WebSphere Application Server ou o servidor JVM mais recente do CICS TS. A segunda versão do terminal da web em serviço é fornecida como um programa de aplicativo do provedor de serviços do CICS (DFH0XODE).

O Arquivo	Descrição
ExampleAppClientV855.war	Cliente de front-end de serviço da web do gerenciador de catálogo
ExampleAppWrapperClientV855.war	Cliente de front-end de serviço da web para wrappers de serviços da web
ExampleAppDispatchOrderV855.war	Aplicativo do provedor de serviços da web de despacho do pedido

Esses WARs foram exportados a partir de projetos dinâmicos da web. Para implementar os arquivos WAR, consulte .

Observe que você precisará ativar o recurso `jax-ws` no servidor Liberty JVM incluindo, por exemplo, `</feature> jaxws-2.2 </feature>` no arquivo de configuração do servidor Liberty, `server.xml`.

Figura 1 mostra uma configuração do aplicativo de exemplo com uma versão do front-end do cliente da web e o provedor de serviços CICS como o terminal de serviço da web de despacho do pedido. Ele também inclui um cliente de serviço da web em um sistema CICS.

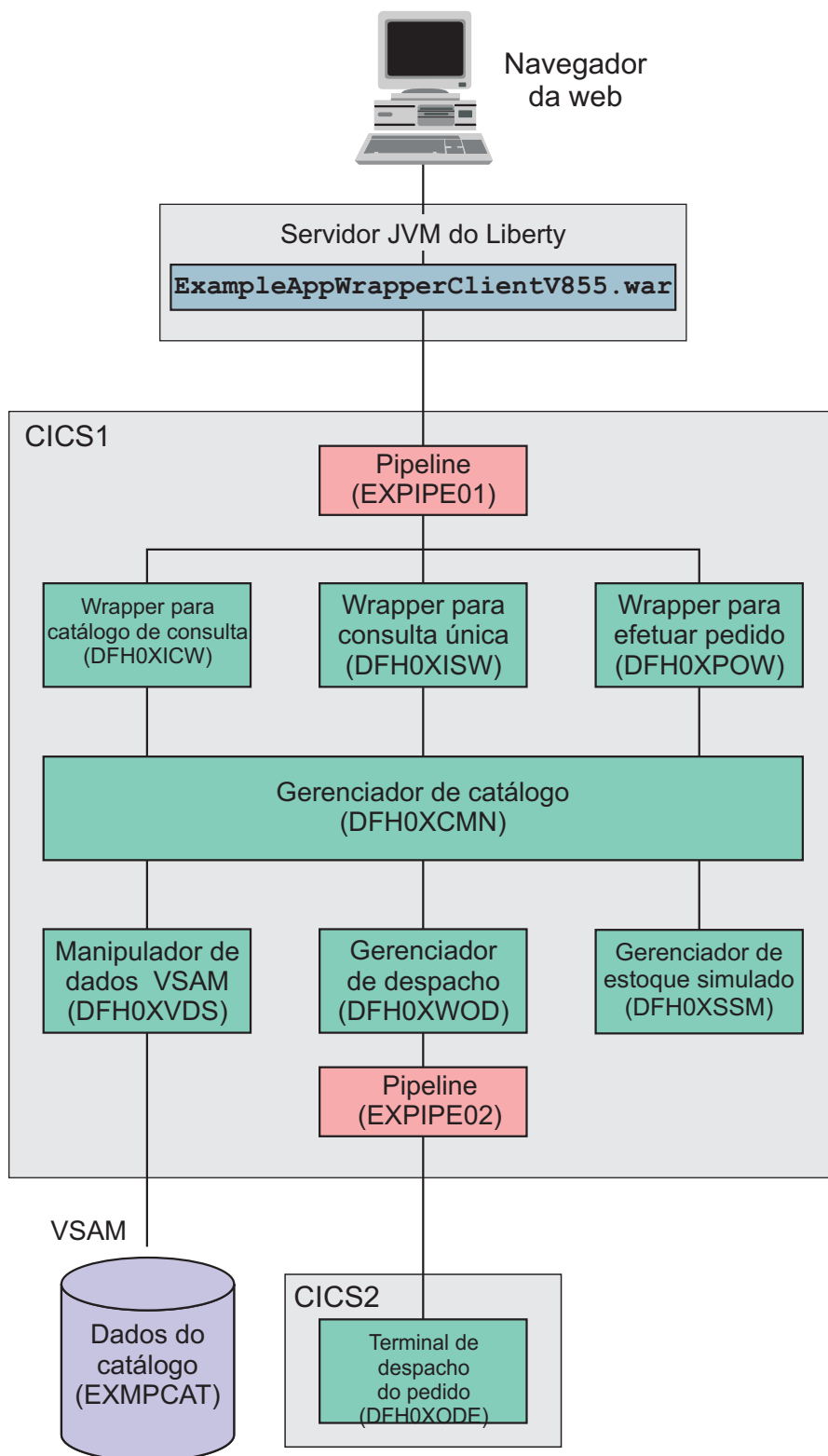


Nesta configuração, o aplicativo é acessado por meio de dois clientes diferentes:

- Um cliente do navegador da web conectado ao servidor Liberty JVM, no qual `ExampleAppClientV855.War` é implementado.

- Cliente de serviço da web do CICS DFH0XECC. Esse cliente usa a mesma lógica de apresentação BMS que o aplicativo base, mas usa o módulo DFH0XCUI em vez de DFH0XGUI.

Figura 2 mostra outra versão de front-end do Web client, com o provedor de serviços CICS como o terminal de serviço da web de despacho do pedido.



Nesta configuração, o cliente do navegador da web é conectado ao servidor Liberty JVM, no qual `ExampleAppWrapperClientV855.war` é implementado. No CICS, três aplicativos de wrapper (para o catálogo de consultas, a consulta única e funções de

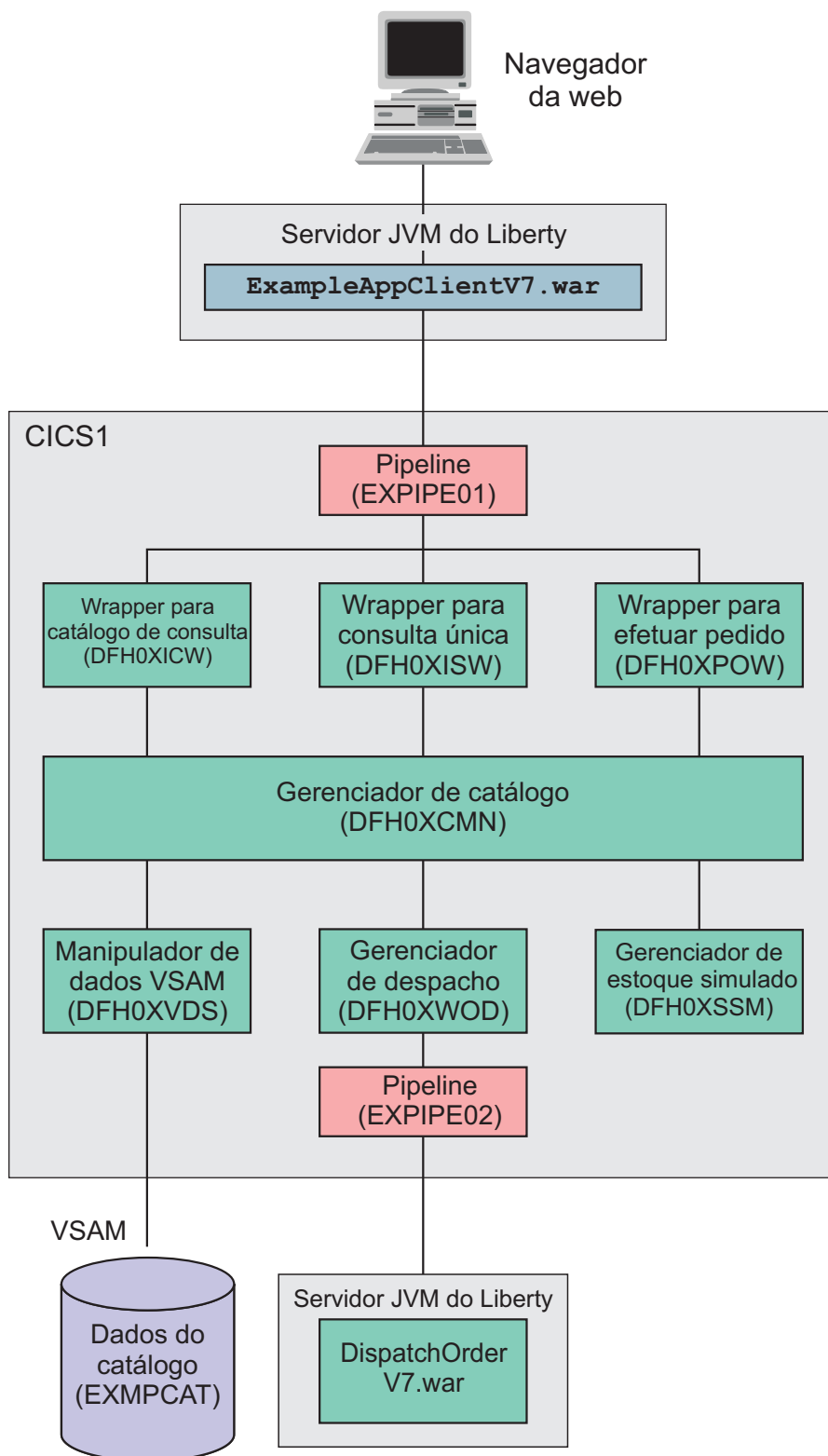
criação de pedido) são implementados como aplicativos do provedor de serviços. Eles, por sua vez, vinculam-se ao aplicativo base.

Para o Dispatch Manager em seu sistema CICS chamar esse terminal, é necessário mudar a configuração a seguir usando a transação de configuração ECFG:

- Outbound WebService? como YES
- Outbound WebService URI para o URI no qual o terminal Dispatch Order está sendo implementado, por exemplo, `http://cics2:8080/exampleApp/dispatchOrder`

Para obter detalhes adicionais sobre como configurar o aplicativo de exemplo, consulte Configurando o aplicativo de exemplo.

Figura 3 mostra uma configuração do aplicativo de exemplo com o front-end do Web client e o terminal de serviço da web de despacho do pedido no servidor Liberty JVM.



Nesta configuração, o cliente do navegador da web está conectado ao servidor Liberty JVM, no qual ExampleAppClientV855.war foi implementado. O terminal de serviço da web de despacho do pedido ExampleAppDispatchOrderV855.war está instalado no servidor Liberty JVM.

Para Dispatch Manager em seu sistema CICS chamar esse terminal, é necessário mudar a configuração a seguir usando a transação de configuração ECFG

- Outbound WebService? como YES
- Outbound WebService URI para o URI no qual o terminal Dispatch Order está sendo implementado, por exemplo, `http://mylibertyserver:9080/ExampleAppDispatchOrderV855/DispatchOrder`

Para obter detalhes adicionais sobre como configurar o aplicativo de exemplo, consulte Configurando o aplicativo de exemplo.

Configurando o Suporte à Página de Códigos

Conforme fornecido, o aplicativo de exemplo usa dois conjuntos de caracteres codificados. Você deve configurar o sistema para ativar a conversão de dados entre os dois conjuntos de caracteres.

Sobre Esta Tarefa

Os conjuntos de caracteres codificados utilizados no aplicativo de exemplo são:

037 EBCDIC Grupo 1: USA, Canadá (z/OS), Países Baixos, Portugal, Brasil, Austrália, Nova Zelândia)

1208 UTF-8 Nível 3

Procedimento

Inclua as instruções a seguir na imagem de conversão para seu sistema z/OS:

```
CONVERSION 037,1208;  
CONVERSION 1208,037;
```

Para obter mais informações, consulte Conversão de dados Unicode pelo z/OS.

Definindo o Cliente de Serviço da Web e Programas Wrappers

Se não estiver utilizando a instalação automática do programa, você deverá definir as definições de recurso para o cliente de serviço da web e programas wrappers.

Sobre Esta Tarefa

Opcional: os usuários que não usam o CICS Explorer podem usar a transação CEDA para corrigir e instalar os recursos contidos nos grupos DFH\$EXBS e DFH\$EXWS.

Procedimento

Utilize o CICS Explorer para definir definições de recurso PROGRAM para os programas wrappers, selecionando **Definições > Definições de Programa**. Clique com o botão direito do mouse na visualização Definições de Programa e selecione **Novo** para criar uma nova definição de programa. Digite um grupo CSD na caixa de texto **Grupo CSD** e digite o nome do programa na caixa de texto **Nome**. Clique em **Concluir** para definir a definição de PROGRAM. Crie definições para os programas COBOL a seguir:

Tabela 17. Membros SDFHSAMP Contendo Código Fonte COBOL para os Módulos do Wrapper

Member name	Descrição
DFH0XECC	Programa cliente de serviços da web

Tabela 17. Membros SDFHSAMP Contendo Código Fonte COBOL para os Módulos do Wrapper (continuação)

Member name	Descrição
DFH0XICW	Programa wrapper para o serviço inquireCatalog.
DFH0XISW	Programa wrapper para o serviço inquireSingle.
DFH0XPOW	Programa wrapper para o serviço purchaseOrder.

Instalando o Suporte de Serviço da Web

Antes de poder executar o suporte de serviço da web para o aplicativo de exemplo, você deve criar dois diretórios z/OS UNIX e criar os recursos do CICS necessários.

Os Diretórios do z/OS UNIX:

O suporte de serviço da web para o aplicativo de exemplo requer um diretório shelf e um diretório de recebimento no z/OS UNIX.

O diretório shelf é usado para armazenar os arquivos de ligação WEBSERVICE que estão associados aos recursos WEBSERVICE. Cada recurso WEBSERVICE é, por sua vez, associado a um PIPELINE. O diretório shelf é gerenciado pelo recurso PIPELINE e você não deve modificar seu conteúdo diretamente. Vários PIPELINES podem usar o mesmo diretório shelf, pois o CICS assegura uma estrutura de diretório exclusiva abaixo do diretório shelf para cada PIPELINE.

O diretório de recebimento é o diretório que contém os arquivos de ligação WEBSERVICE associados a um PIPELINE. Quando um PIPELINE é instalado, ou em resposta a um comando **PERFORM PIPELINE SCAN**, as informações nos arquivos de ligação são usadas para criar dinamicamente as definições de WEBSERVICE e URIMAP associadas ao PIPELINE.

O aplicativo de exemplo usa /var/cicsts para o diretório shelf.

Criando a Definição de Pipeline:

A definição completa de um pipeline consiste em um recurso PIPELINE e um arquivo de configuração de PIPELINE. O arquivo contém os detalhes dos manipuladores de mensagem que agem em solicitações e respostas de serviços da web, à medida que passam pelo pipeline.

Sobre Esta Tarefa

Opcional: os usuários que não usam o CICS Explorer podem usar a transação CEDA para corrigir e instalar os recursos contidos nos grupos DFH\$EXBS e DFH\$EXWS.

O aplicativo de exemplo utiliza o manipulador SOAP 1.1 fornecido para lidar com os envelopes SOAP de solicitações de entrada e de saída. O CICS fornece arquivos de configuração do pipeline de amostra, que você pode utilizar em seu provedor de serviços e solicitante de serviços.

Mais de um serviço da web pode compartilhar um único pipeline, portanto, é necessário definir somente um pipeline para as solicitações de entrada do aplicativo de exemplo. No entanto, você deve definir um segundo pipeline para as

solicitações de saída porque um único pipeline não pode ser configurado para ser um pipeline do provedor e do solicitante ao mesmo tempo.

Se você deseja utilizar pipelines baseados em Java, deverá especificar o arquivo de configuração do provedor de serviços de amostra `basicsoap11javaprovider.xml` em vez de `basicsoap11provider.xml` na etapa 1b. E especifique o arquivo de configuração do solicitante de serviço de amostra `basicsoap11javarequester.xml` em vez de `basicsoap11requester.xml` na etapa 2b. Para obter mais informações sobre arquivos de configuração de amostra, consulte Arquivos de configuração de pipeline. Além disso, se desejar usar o manipulador de aplicativo Axis2 em seu pipeline baseado em Java, você deverá substituir EXPIPE01 por EXPIPE03 na etapa 1a e EXPIPE02 por EXPIPE04 na etapa 2a

Procedimento

1. Use o CICS Explorer para criar uma definição de pipeline para o provedor de serviços.
 - a. Crie uma definição de PIPELINE para os programas wrappers usando o CICS Explorer selecionando **Definições > Definições de Pipeline**. Clique com o botão direito do mouse na visualização Definições de Pipeline e selecione **Novo** para criar uma nova definição de pipeline. Digite DFH\$EXWS na caixa de texto **Grupo de Recursos** e digite EXPIPE01 na caixa de texto **Nome**. Clique em **Concluir** para criar a definição de PIPELINE. Como alternativa, é possível copiar a definição de PIPELINE a partir do grupo fornecido pelo CICS, DFH\$EXWS. Clique com o botão direito do mouse em DFH\$EXWS na visualização Definição do Grupo de Recursos e selecione **Novo > Definição de Pipeline**.
 - b. Dê um clique duplo na definição de PIPELINE e selecione a guia **Atributos** no editor Definição de Pipeline (EXPIPE01). Em **Detalhes**, o **Arquivo de Configuração** deve ser configurado com o local dos arquivos de amostra `/usr/lpp/cicsts/samples/pipelines/basicsoap11provider.xml`, em que `/usr/lpp/cicsts` é o caminho para os arquivos em seu diretório, **Shelf** deve ser `/var/cicsts/`, **Status** deve ser **ENABLED** e **WS Directory** deve ser `/usr/lpp/cicsts/samples/webservices/wsbind/provider/`.

As entradas do z/OS UNIX fazem distinção entre maiúsculas e minúsculas e assumem uma raiz da instalação padrão do CICS z/OS UNIX igual a `/usr/lpp/cicsts`.
2. Use o CICS Explorer para criar uma definição de PIPELINE para o solicitante de serviço.
 - a. Crie uma definição de PIPELINE para os programas wrappers usando o CICS Explorer selecionando **Definições > Definições de Pipeline**. Clique com o botão direito do mouse na visualização Definições de Pipeline e selecione **Novo** para criar uma nova definição de pipeline. Digite DFH\$EXWS na caixa de texto **Grupo de Recursos** e digite EXPIPE02 na caixa de texto **Nome**. Clique em **Concluir** para criar a definição de PIPELINE. Como alternativa, é possível copiar a definição de PIPELINE a partir do grupo fornecido pelo CICS, DFH\$EXWS.
 - b. Dê um clique duplo na definição de PIPELINE e selecione a guia **Atributos** no editor Definição de Pipeline (EXPIPE02). Em **Detalhes**, **Arquivo de Configuração** deve ser configurado com o local dos arquivos de amostra `/usr/lpp/cicsts/samples/pipelines/basicsoap11requester.xml`, em que `/usr/lpp/cicsts` é o caminho para os arquivos em seu diretório, **Shelf** deve ser `/var/cicsts/`, **Status** deve ser **ENABLED** e **WS Directory** deve ser `/usr/lpp/cicsts/samples/webservices/wsbind/requester/`.

Criando um Serviço TCP/IP:

Como o cliente se conecta aos seus serviços da web sobre um transporte HTTP, você deve definir um serviço TCP/IP para receber o tráfego HTTP de entrada.

Procedimento

Use o CICS Explorer para criar uma definição de TCPIPSERVICE para manipular solicitações de HTTP de entrada.

1. Crie uma definição de TCPIPSERVICE selecionando **Definições > Definições de Serviço TCP/IP**. Clique com o botão direito do mouse na visualização Definições de Serviço TCP/IP e selecione **Novo** para criar uma nova definição. Digite DFH\$EXWS na caixa de texto **Grupo de Recursos** e digite EXMPPORT na caixa de texto **Nome**. Você deve especificar um número de porta; digite o número de qualquer porta não utilizada em seu sistema CICS. Clique em **Concluir** para criar a definição de TCPIPSERVICE.
2. Dê um clique duplo na definição de TCPIPSERVICE. Na guia **Atributos** no editor Definição de Serviço TCP/IP (EXMPPORT), configure os atributos a seguir:
 - Urm** deve ser DFHWBAAX
 - Protocol** deve ser HTTP
 - Transaction** deve ser CWXN
3. Use os valores padrão para todos os outros atributos.

Opcional: os usuários que não usam o CICS Explorer podem usar a transação CEDA para corrigir e instalar os recursos contidos nos grupos DFH\$EXBS e DFH\$EXWS.

Instalando Dinamicamente os Recursos WEBSERVICE e URIMAP:

Cada função que é exposta como um serviço da web requer um recurso WEBSERVICE para mapear entre o XML recebido do SOAP BODY e a interface de COMMAREA do programa e um recurso URIMAP que roteie solicitações recebidas para o pipeline e serviço da web corretos. Embora seja possível usar definição de recurso online (RDO) para definir e instalar seus recursos WEBSERVICE e URIMAP, também é possível que o CICS os crie dinamicamente quando um recurso de pipeline é instalado.

Sobre Esta Tarefa

Opcional: os usuários que não usam o CICS Explorer podem usar a transação CEDA para corrigir e instalar os recursos contidos nos grupos DFH\$EXBS e DFH\$EXWS.

Procedimento

1. Use o CICS Explorer para instalar os recursos de PIPELINE.
 - a. Selecione **Definições > Definições de Pipeline**. Clique com o botão direito do mouse na definição de PIPELINE EXPIPE01 na visualização Definições de Pipeline e selecione **Instalar**. Selecione sua região do CICS de destino selecionando a caixa de seleção. Clique em **OK** para instalar o PIPELINE.

Nota: Se você criou definições de pipeline baseadas em Java no “Criando a Definição de Pipeline” na página 661, clique com o botão direito do mouse na definição de PIPELINE EXPIPE03 na visualização Definições de Pipeline.

- b. Repita esse processo para a definição de PIPELINE EXPIPE02 ou EXPIPE04 para pipelines baseadas em Java.

Ao instalar cada recurso PIPELINE, o CICS varre o diretório especificado no atributo PIPELINE WSDIR (o diretório de recebimento). Para cada arquivo de ligação WEBSERVICE no diretório, que é para cada arquivo com o sufixo .wsbind, o CICS instala um recurso WEBSERVICE e um recurso URIMAP se estes recursos não existirem.

O recurso URIMAP fornece ao CICS as informações para associar o recurso WEBSERVICE a um URI específico. Os recursos existentes serão substituídos, se as informações no arquivo de ligação forem mais recentes do que os recursos existentes.

Um segundo recurso URIMAP opcional é instalado se um arquivo WSDL ou archive WSDL foi copiado no diretório de recebimento. Este recurso URIMAP fornece ao CICS as informações para associar o archive WSDL ou o documento WSDL a um URI específico para que os solicitantes externos possam usar o URI para descobrir o archive WSDL ou o documento WSDL.

Quando o PIPELINE é desativado e descartado posteriormente, todos os recursos WEBSERVICE e URIMAP associados também são descartados.

2. Se você já tiver instalado o recurso PIPELINE, utilize o comando **PERFORM PIPELINE SCAN** para iniciar a varredura do diretório de recebimento PIPELINE.

Quando você tiver instalado os recursos PIPELINE, os seguintes recursos WEBSERVICE e os recursos URIMAP associados para o pipeline do provedor serão instalados em seu sistema:

```
dispatchOrder
dispatchOrderEndpoint
inquireCatalog
inquireCatalogClient
inquireCatalogWrapper
inquireSingle
inquireSingleClient
inquireSingleWrapper
placeOrder
placeOrderClient
placeOrderWrapper
```

Os nomes dos recursos WEBSERVICE são derivados dos nomes dos arquivos de ligação WEBSERVICE; os nomes dos recursos URIMAP são gerados dinamicamente. Um URIMAP adicional é gerado para cada documento WSDL que existe no diretório de recebimento do pipeline. É possível visualizar os recursos selecionando **Operações > Serviços da Web** para abrir a visualização Serviços da Web. Clique com o botão direito do mouse no recurso WEBSERVICE e selecione **Abrir Relacionado > Mapa de URI**.

A visualização do CICS Explorer mostra os nomes do recurso PIPELINE, do recurso URIMAP e do programa de destino que está associado a cada serviço da web. Neste exemplo, não há URIMAP ou programa de destino para WEBSERVICE(dispatchOrder) porque o recurso WEBSERVICE é para uma solicitação de saída.

WEBSERVICE(dispatchOrderEndpoint) representa a implementação do CICS local do serviço do pedido de despacho.

Criando os Recursos WEBSERVICE com Definição de Recurso Online (RDO):

Como uma alternativa ao uso do mecanismo de varredura de pipeline para instalar recursos WEBSERVICE, você pode criar e instalá-los utilizando a definição de recurso online (RDO).

Antes de Iniciar

Importante: Se você usar RDO para definir os recursos WEBSERVICE e URIMAP, deverá assegurar que seus arquivos de ligação de serviço da web **não** estejam no diretório de recebimento do PIPELINE. Isto assegura que os recursos WEBSERVICE e URIMAP não sejam instalados dinamicamente durante uma varredura do pipeline do diretório de recebimento. Como alternativa, é possível assegurar que nenhum valor seja especificado para WSDIR no PIPELINE. Entretanto, se você não especificar um valor para WSDIR, nenhuma varredura de pipeline do diretório de recebimento ocorrerá. Portanto, todos os recursos WEBSERVICE e URIMAP precisam ser criados e instalados usando RDO.

Procedimento

1. Use o CICS Explorer para criar uma definição de WEBSERVICE para a função de catálogo de consulta do aplicativo de exemplo.
 - a. Crie uma definição de WEBSERVICE usando o CICS Explorer selecionando **Definições > Definição de Serviço da Web**.
 - b. Clique com o botão direito do mouse na visualização Definições de Serviço da Web e selecione **Novo** para criar uma nova definição de WEBSERVICE.
 - c. Digite DFH\$EXWS na caixa de texto **Grupo de Recursos**, digite EXINQWS na caixa de texto **Nome** e digite EXPIPE01 na caixa de texto **Pipeline** ou digite EXPIPE03 para pipelines baseados em Java. Você deve inserir o atributo WSBind antes de poder criar a definição de WEBSERVICE. Na caixa de texto **Arquivo WSBind**, digite /usr/lpp/cicsts/samples/webservices/wsbind/provider/inquireCatalog.wsbind.
 - d. Clique em **Concluir** para criar a definição de WEBSERVICE.
2. Repita a etapa precedente para cada uma das funções a seguir do aplicativo de exemplo.

Função	Nome de WEBSERVICE	atributo PIPELINE	Atributo WSBind
INQUIRE SINGLE ITEM	EXINQWS	EXPIPE01 ou EXPIPE03	/usr/lpp/cicsts/samples/webservices/wsbind/provider/inquireSingle.wsbind
PLACE ORDER	EXORDRWS	EXPIPE01 ou EXPIPE03	/usr/lpp/cicsts/samples/webservices/wsbind/provider/placeOrder.wsbind
DISPATCH STOCK	EXODRQWS	EXPIPE02 ou EXPIPE04	/usr/lpp/cicsts/samples/webservices/wsbind/requester/dispatchOrder.wsbind
Terminal DISPATCH STOCK (opcional)	EXODEPWS	EXPIPE01 ou EXPIPE03	/usr/lpp/cicsts/samples/webservices/wsbind/provider/dispatchOrderEndpoint.wsbind

Criando os Recursos URIMAP com Definição de Recurso Online (RDO):

Como uma alternativa ao uso do mecanismo de varredura de pipeline para instalar recursos URIMAP, você pode criar e instalá-los utilizando a definição de recurso online (RDO).

Antes de Iniciar

Importante: Se você usar RDO para definir os recursos WEBSERVICE e URIMAP, deverá assegurar que seus arquivos de ligação de serviço da web **não** estejam no diretório de recebimento do PIPELINE. Isto assegura que os recursos WEBSERVICE e URIMAP não sejam instalados dinamicamente durante uma varredura do pipeline do diretório de recebimento. Como alternativa, é possível assegurar que nenhum valor seja especificado para WSDIR no PIPELINE. Entretanto, se você não especificar um valor para WSDIR, nenhuma varredura de pipeline do diretório de recebimento ocorrerá. Portanto, todos os recursos WEBSERVICE e URIMAP precisam ser criados e instalados usando RDO.

Procedimento

1. Use o CICS Explorer para criar uma definição de URIMAP para a função de catálogo de consulta do aplicativo de exemplo.
 - a. Crie uma definição de URIMAP no CICS Explorer selecionando **Definições > Definição de Mapa de URI**.
 - b. Clique com o botão direito do mouse na visualização Definições de Mapa de URI e selecione **Novo** para criar uma nova definição de URIMAP.
 - c. Digite INQCURI na caixa de texto **Nome** e digite * na caixa de texto **Host**. Você deve inserir o atributo **Caminho** antes de poder criar as definições de URIMAP. Na caixa de texto **Caminho**, digite /exampleApp/inquireCatalog. **Uso** deve ser configurado como **Pipeline**; o recurso PIPELINE é EXPIPE01 ou EXPIPE03 para pipelines baseados em Java.
 - d. Clique em **Concluir** para concluir a definição de URIMAP.
 - e. Dê um clique duplo no novo recurso URIMAP para abrir o Editor. Na guia **Atributos** no Editor, configure o atributo **Serviço da Web** como EXINQCWS e **Serviço TCP/IP** como S0APP0RT.
2. Repita a etapa anterior para cada uma das funções restantes do aplicativo de exemplo. Use os nomes a seguir para seus URIMAPs:

Função	Nome de URIMAP
INQUIRE SINGLE ITEM	INQSURI
PLACE ORDER	ORDRURI
DISPATCH STOCK	Não requerido
Terminal DISPATCH STOCK (opcional)	ODEPURI

3. Especifique os atributos distintos a seguir para cada URIMAP:

Função	Nome de URIMAP	PATH	WEBSERVICE
INQUIRE SINGLE ITEM	INQSURI	/exampleApp/inquireSingle	EXINQSWs
PLACE ORDER	ORDRURI	/exampleApp/placeOrder	EXORDRWS
Terminal DISPATCH STOCK (opcional)	ODEPURI	/exampleApp/dispatchOrder	EXODEPWS

Concluindo a Instalação:

Para concluir a instalação, instale o grupo RDO que contém suas definições de recurso.

Procedimento

Clique com o botão direito do mouse no grupo de recursos na janela Definições do Grupo de Recursos. Selecione **Instalar**. Certifique-se que seu CICSplex esteja correto e que você selecionou sua região de destino, em seguida, clique em **OK**.

Resultados

Seu RDO agora está instalado e o aplicativo está pronto para uso.

Opcional: os usuários que não usam o CICS Explorer podem usar a transação CEDA para corrigir e instalar os recursos contidos nos grupos DFH\$EXBS e DFH\$EXWS.

Configurando o Web Client

Antes de poder usar o cliente ad web, deve-se implementar o arquivo Java da web (WAR) para o frontend do cliente da web client em um dos ambientes suportados e configurá-lo para chamar os terminais apropriados no sistema CICS.

Sobre Esta Tarefa

Os ambientes a seguir são suportados para as duas versões do aplicativo de front-end do Web client, ExampleAppClientV855.war e ExampleAppWrapperClientV855.war:

- Servidor CICS Liberty JVM com o Perfil WebSphere Liberty mais recente

Os arquivos WAR estão localizados no diretório `hlq/samples/webservices/client` no z/OS UNIX.

Procedimento

1. Para iniciar o Web client, insira a URL a seguir em seu navegador da web, em que *mylibertyserver* é o nome do host do servidor Liberty JVM no qual o Web client está instalado.
 - Para ExampleAppClientV855.war, use a URL `http://mylibertyserver:9080/ExampleAppClientV855/`
 - Para ExampleAppWrapperClientV855.war, use a URL `http://mylibertyserver:9080/ExampleAppWrapperClientV855/`

O aplicativo de exemplo exibe a página a seguir:

CICS Example - Catalog Application	
<div>LIST ITEMS</div> <div>INQUIRE</div> <div>ORDER ITEM</div> <div>CONFIGURE</div>	<p>Welcome to the CICS Catalog Example Application</p> <p>Please select an option from the menu</p>
CICS Transaction Server for z/OS	

2. Clique em **CONFIGURAR** para exibir a página de configuração. A página de configuração é exibida.
3. Insira o novo terminal para o serviço da web do catálogo de Consulta, Consulte o item e Faça o pedido.
 - a. Nas URLs, substitua a sequência `myCicsServer` pelo nome do sistema no qual seu CICS está em execução.
 - b. Substitua o número da porta 8080 pelo número da porta configurado no recurso de definição TCPIP SERVICE.
4. Clique em **SUBMIT**.

Resultados

O aplicativo da web agora está pronto para execução.

O que Fazer Depois

A URL para a chamada de serviço da web é armazenada em uma sessão HTTP. Portanto, é necessário repetir esta etapa de configuração toda vez que um navegador da web é conectado pela primeira vez ao cliente.

Executando o Aplicativo Ativado para Serviço da Web

É possível chamar o aplicativo de exemplo a partir de um navegador da web.

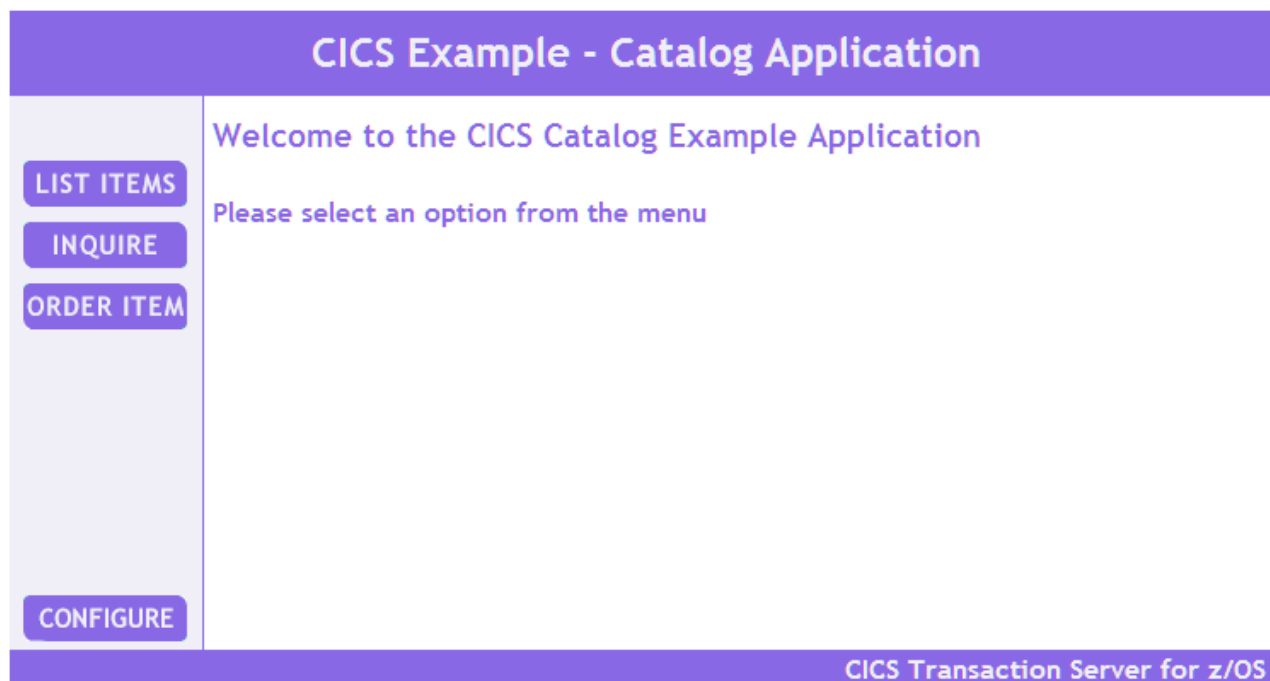
Sobre Esta Tarefa

Certifique-se de que você configurou o Web client antes de continuar. Consulte Configurando o aplicativo de exemplo

Procedimento

1. Insira a URL a seguir em seu navegador da web: `http://mylibertyserver:9080/ExampleAppClientV855/`, em que *mylibertyserver* é o

nome do host do servidor no qual o cliente de serviço da web está instalado. O aplicativo de exemplo exibe a página a seguir:



2. Clique no botão **INQUIRE**. O aplicativo de exemplo exibe a página a seguir:

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. Insira um número de item e clique no botão **SUBMIT**.

Dica: O aplicativo base é iniciado com números de itens na sequência 0010, 0020,... até 0210.

O aplicativo exibe a página a seguir, que contém uma lista de itens no catálogo, iniciando com o número do item inserido.

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. Selecione o item que deseja solicitar.
 - a. Clique no botão de opções na coluna **Select** para obter o item que deseja solicitar.
 - b. Clique no botão **SUBMIT**.
- O aplicativo exibe a página a seguir:

CICS Example - Catalog Application

LIST ITEMS

INQUIRE

BACK

CONFIGURE

Enter Order Details

Item Reference Number

0040

Quantity

001

User Name

AUSER

Department Name

CICS1

SUBMIT

CICS Transaction Server for z/OS

5. Para fazer um pedido, insira as informações a seguir.
- Preencha o campo **Quantity**. Especifique o número de itens que deseja solicitar.
 - Preencha o campo **User Name**. Insira uma sequência de 1 a 8 caracteres. O aplicativo base não verifica o valor que é inserido aqui.
 - Preencha o campo **Department Name**. Insira uma sequência de 1 a 8 caracteres. O aplicativo base não verifica o valor que é inserido aqui.
 - Clique no botão **SUBMIT**.

O aplicativo exibe a página a seguir, para confirmar se o pedido foi feito:

CICS Example - Catalog Application	
	Order Placed
LIST ITEMS	ORDER SUCESSFULLY PLACED
INQUIRE	
ORDER ITEM	
BACK	
CONFIGURE	
CICS Transaction Server for z/OS	

Implementando o Aplicativo de Exemplo

É possível usar o assistente de serviços da web para implementar partes do aplicativo de exemplo como um serviço da web. Embora o aplicativo funcione sem executar esta tarefa, você deve executar uma tarefa semelhante se desejar implementar seus próprios aplicativos para estender o aplicativo de exemplo.

Extraíndo a Interface do Programa

Para implementar um programa com o assistente de serviços da web do CICS, você deverá criar um copybook que corresponda à interface COMMAREA ou do contêiner.

Sobre Esta Tarefa

Neste exemplo, a função INQUIRE SINGLE ITEM do programa do gerenciador de catálogo central (DFH0XCMN) é implementada como um serviço da web. A interface para este programa é uma COMMAREA; a estrutura da COMMAREA é definida no copybook DFH0XCP1:

```
* Catalogue COMMAREA structure
   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).
* Fields used in Inquire Catalog
   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).
*   Fields used in Inquire Single
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 em Place Order
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).

```

O copybook define três interfaces separadas para as funções INQUIRE CATALOG, INQUIRE SINGLE ITEM e PLACE ORDER, que são sobrepostas umas nas outras no copybook. No entanto, o utilitário DFHLS2WS não suporta a instrução REDEFINES. Portanto, você deve extrair do copybook combinado somente aquelas seções que se relacionam à função única de consulta:

```

*   Catalogue COMMAREA structure
03 CA-REQUEST-ID            PIC X(6).
03 CA-RETURN-CODE           PIC 9(2) DISPLAY.
03 CA-RESPONSE-MESSAGE      PIC X(79).
*   Fields used in Inquire Single
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).

```

O elemento redefinido CA-REQUEST-SPECIFIC foi removido e substituído pela seção do copybook que o redefiniu para a função única de consulta. O copybook agora está adequado para uso com o assistente de serviços da web.

O copybook é fornecido com o aplicativo de exemplo como o copybook DFH0XCP4.

Executando o Programa do Assistente de Serviço da Web DFHLS2WS

O assistente de serviços da web do CICS consiste em dois programas em lote que podem ajudá-lo a transformar aplicativos CICS existentes em serviços da web e a ativar aplicativos CICS para usarem serviços da web fornecidos pelos provedores externos. Programa DFHLS2WS transforma uma estrutura de linguagem para gerar um arquivo de ligação de serviço da web e uma descrição de serviços da web.

Procedimento

1. Copie a JCL de amostra fornecida em um arquivo de trabalho adequado. A JCL é fornecida em `samples/webservices/JCL/LS2WS`.
2. Inclua uma placa JOB válida na JCL.
3. Codifique os parâmetros para DFHLS2WS. Os parâmetros a seguir são necessários para a função INQUIRE SINGLE ITEM do aplicativo de exemplo:

```
//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/wsdlinquireSingle.wsdlinquireSingle  
*/
```

LOGFILE=/u/exampleapp/wsbinding/inquireSingle.log

O arquivo que é usado para registrar informações de diagnóstico do DFHLS2WS. O arquivo normalmente é usado somente pela organização de suporte ao software IBM.

PDSLIB=CICSHLQ.SDFHSAMP

O nome do conjunto de dados particionados (PDS) no qual o assistente de serviços da web procura copybooks que definem as estruturas de solicitação e resposta. No exemplo, está é o conjunto de dados SDFHSAMP instalado pelo CICS.

REQMEM=DFH0XCP4

RESPMEM=DFH0XCP4

Estes parâmetros definem a estrutura de linguagem para a solicitação e a resposta para o programa. No exemplo, a solicitação e a resposta possuem a mesma estrutura e são definidas pelo mesmo copybook.

LANG=COBOL

O programa de destino e as estruturas de dados são gravados em COBOL.

PGMNAME=DFH0XCMN

O nome do programa de destino que é iniciado quando uma solicitação de serviço da web é recebida.

PGMINT=COMMAREA

O programa de destino é chamado com uma interface COMMAREA.

URI=mycicsserver:myport/exampleApp/inquireSingle

A parte exclusiva do URI que é usada na definição de serviço da web gerada e usada para criar o recurso URIMAP que mapeia solicitações recebidas para o serviço da web correto. O valor especificado resulta na disponibilidade do serviço para clientes externos em:

`http://mycicsserver:myport/exampleApp/inquireSingle`

em que *mycicsserver* e *myport* são o endereço do servidor CICS e a porta na qual este recurso WEBSERVICE foi instalado.

Nota: O parâmetro não possui um '/' inicial.

WSBIND=/u/exampleapp/wsbind/inquireSingle.wsbind

O local no z/OS UNIX no qual o arquivo de ligação de serviço da web é gravado.

Nota: Se o arquivo precisar ser usado com o mecanismo de varredura de pipeline, ele deverá ter a extensão .wsbind.

WSDL=/u/exampleapp/wsd1/inquireSingle.wsd1

O local no z/OS UNIX no qual o arquivo contendo a descrição do serviço da web gerado é gravado. É uma boa prática usar nomes correspondentes para o arquivo de ligação de serviço da web e sua descrição de serviços da web correspondente.

De modo convencional, arquivos contendo descrições de serviços da web possuem a extensão .wsdl.

A descrição de serviços da web fornece as informações que um cliente deve usar para acessar o serviço da web. Ela contém uma definição de esquema XML da solicitação e resposta e informações de local para o serviço.

4. Execute a tarefa. Uma descrição de serviços da web e um arquivo de ligação de serviço da web são criados nos locais especificados.

Um Exemplo do Documento WSDL Gerado

Um exemplo do documento de descrição de serviços da web (WSDL) que é gerado quando o programa DFHLS2WS do assistente de serviços da web é executado.

```
<?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 TCPIP 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>
```

Implementando o Arquivo de Ligação de Serviços da Web

O arquivo de ligação WEBSERVICE, criado por DFHLS2WS, é implementado em sua região do CICS dinamicamente quando você instala um recurso PIPELINE.

Sobre Esta Tarefa

Quando um comando de varredura de pipeline é emitido, o CICS varre o diretório de recebimento para procurar arquivos de ligação WEBSERVICE com a extensão .wsbind. Para cada arquivo de ligação localizado, o CICS determina se deve instalar um recurso WEBSERVICE.

Um recurso URIMAP também é criado para mapear o URI, conforme fornecido na JCL, para o recurso WEBSERVICE instalado e o PIPELINE no qual o serviço da web é instalado. Quando um recurso WEBSERVICE varrido é descartado, o recurso URIMAP associado a ele também é descartado.

Procedimento

1. Modifique a definição de PIPELINE para seu pipeline do provedor PIPELINE(EXPIPE01) no CICS Explorer selecionando **Definições > Definições de Pipeline**. Dê um clique duplo em EXPIPE01 para abrir o editor Definição de Pipeline (EXPIPE01). Na guia **Atributos**, altere o parâmetro **WS Directory** para

/u/exampleapp/wsbind. Este diretório de recebimento contém o arquivo de ligação WEBSERVICE que você gerou com DFHLS2WS.

2. Copie qualquer outro arquivo de ligação WEBSERVICE usado pelo aplicativo no mesmo diretório. Neste exemplo, os arquivos a seguir são copiados:

inquireCatalog

placeOrder

Eles são fornecidos no diretório /usr/lpp/cicsts/samples/webservices/wsbind/provider.

3. Instale o recurso PIPELINE.

Resultados

O CICS cria dois recursos URIMAP; a primeira definição de URIMAP é requerida em um provedor de serviços quando ela contém informações que mapeiam o URI de uma solicitação de serviço da web de entrada para os outros recursos (tal como o recurso PIPELINE) que atendem a solicitação. O segundo URIMAP contém informações que mapeiam o URI de uma solicitação de entrada para o documento ou documentos WSDL associados ao serviço da web.

Componentes do Aplicativo Base

Utilize estas tabelas para entender os componentes do aplicativo base e os membros fornecidos na amostra SDFHSAMP. Os membros SDFHSAMP listados contêm mapas BMS, origem COBOL e copybooks para o aplicativo base, o aplicativo cliente de serviço da web e os módulos de wrapper.

Tabela 18. Membros SDFHSAMP Contendo mapas BMS

Member name	Descrição
DFH0XS1	Macros BMS para o conjunto de mapas que consiste no mapa (EXMENU) para a tela Menu Principal e no mapa (EXORDR) para a tela Detalhes do seu Pedido .
DFH0XS2	Macros BMS para o conjunto de mapas que consiste no mapa (EXINQC) para a tela Catálogo de Consulta .
DFH0XS3	Macros BMS para o conjunto de mapas que consiste no mapa (EXCONF) para a tela Configurar Aplicativo de Catálogo de Exemplo do CICS .
DFH0XM1	Copybook COBOL gerado montando DFH0XS1. DFH0XGUI e DFH0XCUI incluem esse copybook
DFH0XM2U	Copybook COBOL gerado montando DFH0XS2 e editando o resultado para incluir uma estrutura de matriz indexada para facilitar a programação de copybook. DFH0XGUI e DFH0XCUI incluem este copybook.
DFH0XM3	Copybook COBOL gerado montando DFH0XS3. DFH0XCFG inclui este copybook

Tabela 19. Membros SDFHSAMP Contendo Origem COBOL para o Aplicativo Base

Member name	Descrição
DFH0XCFG	Programa chamado pela transação ECFG para ler e atualizar o arquivo de configuração do VSAM.
DFH0XCMN	Programa controlador para o aplicativo de catálogo. Todas as solicitações passam pelo programa controlador.

Tabela 19. Membros SDFHSAMP Contendo Origem COBOL para o Aplicativo Base (continuação)

Member name	Descrição
DFH0XGUI	Programa chamado pela transação EGUI para gerenciar o envio dos mapas BMS para o usuário do terminal e o recebimento dos mapas do usuário do terminal. Este programa se vincula ao programa DFH0XCMN.
DFH0XODE	Uma das duas versões do terminal para o serviço da web de envio do pedido. Esta é a versão que é executada no CICS. Este programa configura o texto "Order in dispatch" na COMMAREA de retorno.
DFH0XSDS	Uma versão <i>em stub</i> ou simulada do programa de armazenamento de dados que permite ao aplicativo trabalhar quando o arquivo de catálogo do VSAM não tiver sido configurado. DFH0XSDS usa dados definidos no programa em vez de dados armazenados em um arquivo VSAM.
DFH0XSOD	Uma versão em stub do programa de envio do pedido. Ele configura o código de retorno na COMMAREA como 0 e retorna ao seu responsável pela chamada. DFH0XSOD é usado quando serviços da web de saída não são necessários.
DFH0XSSM	Uma versão em stub do programa gerenciador de estoque (reabastecimento). DFH0XSSM configura o código de retorno na COMMAREA como 0 e retorna ao seu responsável pela chamada.
DFH0XVDS	A versão de VSAM do programa de armazenamento de dados. DFH0XVDS acessa o arquivo VSAM para executar leituras e atualizações do catálogo.
DFH0XWOD	A versão de serviço da web do programa de despacho do pedido. DFH0XWOD emite um EXEC CICS INVOKE WEBSERVICE para fazer uma chamada de serviço da web de saída para um dispatcher de pedido.

Tabela 20. Membros SDFHSAMP Contendo Copybooks COBOL para o Aplicativo Base

Member name	Descrição
DFH0XCP1	Define uma estrutura COMMAREA que inclui a solicitação e a resposta para o catálogo de consulta, a consulta única e funções de criação de pedido. Programas DFH0XCMN, DFH0XCUI, DFH0XECC, DFH0XGUI, DFH0XICW, DFH0XISW, DFH0XPOW, DFH0XSDS e DFH0XVDS incluem este copybook.
DFH0XCP2	Define uma estrutura COMMAREA para o dispatcher de pedido e módulos do gerenciador de estoque. Programas DFH0XCMN, DFH0XSOD, DFH0XSSM e DFH0XWOD incluem este copybook
DFH0XCP3	Define uma estrutura de dados para uma solicitação e resposta do catálogo de consulta. Usado como entrada para DFHLS2WS para produzir inquireCatalog.wsdl e inquireCatalog.wsbind.
DFH0XCP4	Define uma estrutura de dados para uma solicitação e resposta de consulta única. Usado como entrada para DFHLS2WS para produzir inquireSingle.wsdl e inquireSingle.wsbind.
DFH0XCP5	Define uma estrutura de dados para uma solicitação e resposta de criação de pedido. Usado como entrada para DFHLS2WS para produzir placeOrder.wsdl e placeOrder.wsbind.
DFH0XCP6	Define uma estrutura de dados para uma solicitação e resposta de pedido de despacho. Usado como entrada para DFHLS2WS para produzir dispatchOrder.wsdl e dispatchOrder.wsbind.

Tabela 20. Membros SDFHSAMP Contendo Copybooks COBOL para o Aplicativo Base (continuação)

Member name	Descrição
DFH0XCP7	Define a estrutura de dados para uma solicitação de pedido de despacho. Os programas DFH0XODE e DFH0XWOD incluem este copybook
DFH0XCP8	Define a estrutura de dados para uma resposta do pedido de despacho. Os programas DFH0XODE e DFH0XWOD incluem este copybook.

Tabela 21. Membros SDFHSAMP Contendo o Código Fonte COBOL para o Aplicativo Cliente de Serviço da Web que é Executado no CICS

Member name	Descrição
DFH0XCUI	Programa chamado pela transação ECLI para gerenciar o envio dos mapas BMS para o usuário do terminal e o recebimento dos mapas do usuário do terminal. Ele se vincula ao programa DFH0XECC.
DFH0XECC	Faz solicitações de serviço da web de saída para o aplicativo base, utilizando o comando EXEC CICS INVOKE WEBSERVICE. O serviço da web especificado é um dos seguintes: inquireCatalogClient inquireSingleClient placeOrderClient

Tabela 22. Membros SDFHSAMP Contendo Copybooks COBOL para o Aplicativo Cliente de Serviço da Web que é Executado no CICS. Eles são todos gerado pelo DFHWS2LS e são incluídos pelo programa DFH0XECC.

Member name	Descrição
DFH0XCPA	Define a estrutura de dados para a solicitação do catálogo da consulta.
DFH0XCPB	Define a estrutura de dados para a resposta do catálogo da consulta.
DFH0XCPC	Define a estrutura de dados para a solicitação única da consulta.
DFH0XCPD	Define a estrutura de dados para a resposta única da consulta.
DFH0XCPE	Define a estrutura de dados para a solicitação de realização do pedido.
DFH0XCPF	Define a estrutura de dados para a resposta da realização do pedido.

Tabela 23. Membros SDFHSAMP Contendo Código Fonte COBOL para os Módulos do Wrapper

Member name	Descrição
DFH0XECC	Programa cliente de serviços da web
DFH0XICW	Programa wrapper para o serviço inquireCatalog.
DFH0XISW	Programa wrapper para o serviço inquireSingle.
DFH0XPOW	Programa wrapper para o serviço purchaseOrder.

Tabela 24. Membros SDFHSAMP Contendo Copybooks COBOL para os Módulos do Wrapper

Member name	Descrição
DFH0XWC1	Define a estrutura de dados para a solicitação do catálogo da consulta. O programa DFH0XICW inclui este copybook.
DFH0XWC2	Define a estrutura de dados para a resposta do catálogo da consulta. O programa DFH0XICW inclui este copybook.
DFH0XWC3	Define a estrutura de dados para a solicitação única da consulta. O programa DFH0XISW inclui este copybook.
DFH0XWC4	Define a estrutura de dados para a resposta única da consulta. O programa DFH0XISW inclui este copybook.
DFH0XWC5	Define a estrutura de dados para a solicitação de realização do pedido. O programa DFH0XPOW inclui este copybook.
DFH0XWC6	Define a estrutura de dados para a resposta da realização do pedido. O programas DFH0XPOW inclui este copybook

Tabela 25. Definições do Recurso do CICS

Nome do recurso	Tipo de Recurso	Comentário
EXAMPLE	Grupo de definições de recursos do CICS	Definições de recursos do CICS necessárias para o aplicativo de exemplo.
EGUI	TRANSACTION	Transação para chamar o programa DFH0XGUI para iniciar a interface BMS para o aplicativo (Customizável).
ECFG	TRANSACTION	Transação para chamar o programa DFH0XCFG para iniciar a interface BMS de configuração de exemplo (Customizável).
EXMPCAT	FILE	Definição de arquivo do arquivo EXMPCAT VSAM para o catálogo de aplicativos (Customizável).
EXMPCONF	FILE	Definição de arquivo do arquivo de configuração de aplicativo EXMPCONF.

O Programa do Gerenciador de Catálogo

O gerenciador de catálogo é o programa de controle para a lógica de negócios do aplicativo de exemplo e todas as interações com o aplicativo de exemplo passam por ele.

Para assegurar que a lógica do programa seja simples, a verificação de tipo e a recuperação de erro que o gerenciador de catálogo executa são limitadas.

O gerenciador de catálogo suporta várias operações. Os parâmetros de entrada e saída para cada operação são definidos em uma única estrutura de dados, que é transmitida para o programa e a partir dele em uma COMMAREA.

Estruturas COMMAREA:

Os dados são transmitidos entre o cliente de amostra e os programas do servidor usando uma áreas de comunicação do CICS padrão (COMMAREA).

O fragmento de código a seguir mostra a estrutura COMMAREA do aplicativo gerenciador de catálogo.

```
*   Catalogue COMMAREA structure
      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).
*   Fields used in Inquire Catalog
      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).
*   Fields used in Inquire Single
      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 em Place Order
      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).

*   Dispatcher/Stock Manager COMMAREA structure
      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).
*   Fields used in Dispatcher
      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).
*   Fields used in Stock Manager
      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 operação do Catalog Manager pode retornar um número de códigos de retorno.

Tabela 26. Códigos de Retorno do Gerenciador de Catálogo

Tipo	Código	Explicação
Geral	00	Função concluída sem erro
Arquivo de Catálogo	20	Referência do item não localizada
	21	Erro ao abrir, ler ou terminar navegação do arquivo de catálogo
	22	Erro ao atualizar arquivo
Arquivo de Configuração	50	Erro ao abrir arquivo de configuração
	51	O tipo de armazenamento de dados não era STUB nem VSAM
	52	Comutador de serviço da web de saída não era Y nem N
Serviço da web remoto	30	O comando EXEC CICS INVOKE WEBSERVICE retornou uma condição INVREQ
	31	O comando EXEC CICS INVOKE WEBSERVICE retornou uma condição NOTFND
	32	O comando EXEC CICS INVOKE WEBSERVICE retornou uma condição diferente de INVREQ ou NOTFND
Aplicativo	97	Estoque insuficiente para concluir pedido
	98	A quantidade de pedido não era um número positivo
	99	DFH0XCMN recebeu uma COMMAREA na qual o campo CA-REQUEST-ID não foi configurado com um dos seguintes: 01INQC, 01INQS ou 01ORDR

Operação INQUIRE CATALOG:

Essa operação retorna uma lista de até 15 itens de catálogos, começando com o item especificado pelo responsável pela chamada.

Parâmetros de entrada

CA-REQUEST-ID

Uma sequência que identifica a operação. Para o comando INQUIRE CATALOG, a sequência contém 01INQC.

CA-LIST-START-REF

O número de referência do primeiro item a ser retornado.

Parâmetros de Saída**CA-RETURN-CODE**

Uma sequência que identifica a operação.

CA-RESPONSE-MESSAGE

Uma sequência legível, contendo *num* ITEMS RETURNED em que *num* é o número de itens retornados.

CA-LAST-ITEM-REF

O número de referência do último item retornado.

CA-ITEM-COUNT

O número de itens retornados.

CA-CAT-ITEM

Uma matriz contendo a lista de itens do catálogo retornados. A matriz possui 15 elementos; se menos de 15 itens forem retornados, os elementos restantes da matriz irão conter espaços em branco.

Operação INQUIRE SINGLE ITEM:

Esta operação retorna um único item de catálogo especificado pelo responsável pela chamada.

Parâmetros de entrada**CA-REQUEST-ID**

Uma sequência que identifica a operação. Para o comando INQUIRE SINGLE ITEM, a sequência contém 01INQS.

CA-ITEM-REF-REQ

O número de referência do item a ser retornado.

Parâmetros de Saída**CA-RETURN-CODE**

Uma sequência que identifica a operação.

CA-RESPONSE-MESSAGE

Uma sequência legível, contendo RETURNED ITEM: REF=*item-reference* em que *item-reference* é o número de referência do item devolvido.

CA-SINGLE-ITEM

Uma matriz contendo em seu primeiro elemento o item de catálogo retornado.

Operação PLACE ORDER:

Essa operação faz um pedido para um único item. Se a quantidade necessária não estiver disponível, uma mensagem será retornada para o usuário. Se o pedido for bem-sucedido, será feita uma chamada para o Stock Manager, informando-o qual item foi pedido e a quantidade pedida.

Parâmetros de entrada**CA-REQUEST-ID**

Uma sequência que identifica a operação. Para a operação PLACE ORDER, a sequência contém 01ORDR.

CA-USERID

Um ID do usuário de 8 caracteres que o aplicativo usa para despacho e faturamento.

CA-CHARGE-DEPT

Um ID de departamento de 8 caracteres que o aplicativo usa para despacho e faturamento.

CA-ITEM-REF-NUMBER

O número de referência do item a ser solicitado.

CA-QUANTITY-REQ

O número de itens necessários.

Parâmetros de Saída**CA-RETURN-CODE**

Uma sequência que identifica a operação.

CA-RESPONSE-MESSAGE

Uma sequência legível, contendo ORDER SUCCESSFULLY PLACED.

Operação DISPATCH STOCK:

Esta operação coloca uma chamada no programa de dispatcher do estoque, que por sua vez despacha o pedido ao cliente.

Parâmetros de entrada**CA-ORD-REQUEST-ID**

Uma sequência que identifica a operação. Para a operação DISPATCH ORDER, a sequência contém 01DSP0.

CA-ORD-USERID

Um ID do usuário de 8 caracteres que o aplicativo usa para despacho e faturamento.

CA-ORD-CHARGE-DEPT

Um ID de departamento de 8 caracteres que o aplicativo usa para despacho e faturamento.

CA-ORD-ITEM-REF-NUMBER

O número de referência do item a ser solicitado.

CA-ORD-QUANTITY-REQ

O número de itens necessários.

Parâmetros de Saída**CA-ORD-RETURN-CODE**

Uma sequência que identifica a operação.

Operação NOTIFY STOCK MANAGER:

Esta operação obtém detalhes da ordem que foi estabelecida para decidir se o reabastecimento de estoque é necessário.

Parâmetros de entrada**CA-ORD-REQUEST-ID**

Uma sequência que identifica a operação. Para a operação NOTIFY STOCK MANAGER, a sequência contém 01STK0.

CA-STK-ITEM-REF-NUMBER

O número de referência do item a ser solicitado.

CA-STK-QUANTITY-REQ

O número de itens necessários.

Parâmetros de Saída**CA-ORD-RETURN-CODE**

Uma sequência que identifica a operação.

Estruturas e Definições do Arquivo

O aplicativo de exemplo usa dois arquivos VSAM: o arquivo de catálogo, que contém os detalhes de todos os itens estocados e seus níveis de estoque e o arquivo de configuração, que contém as opções selecionadas pelo usuário para o aplicativo.

Arquivo de Catálogo

O arquivo de catálogo é um arquivo KSDS VSAM que contém todas as informações relacionadas ao inventário do produto.

Registros de Arquivo de Catálogo

Os registros no arquivo têm a seguinte estrutura:

Nome	Tipo de Dados COBOL	Descrição
WS-ITEM-REF-NUM	PIC 9(4)	Número de referência do item
WS-DESCRIPTION	PIC X(40)	Descrição do item
WS-DEPARTMENT	PIC 9(3)	Número de identificação do departamento
WS-COST	PIC ZZZ.99	Preço do item
WS-IN-STOCK	PIC 9(4)	Número de itens em estoque
WS-ON-ORDER	PIC 9(3)	Número de itens no pedido

Arquivo de configuração

O arquivo de configuração é um arquivo KSDS VSAM que contém informações usadas para configurar o aplicativo de exemplo.

Registros do Arquivo de Configuração

O arquivo de configuração é um arquivo KSDS VSAM com quatro registros distintos.

Tabela 27. Registro de Informações Gerais

Nome	Tipo de Dados COBOL	Descrição
PROGS-KEY	PIC X(9)	O campo-chave para o registro de informações gerais, contendo EXMP-CONF.
preenchimento	PIC X	

Tabela 27. Registro de Informações Gerais (continuação)

Nome	Tipo de Dados COBOL	Descrição
DATASTORE	PIC X(4)	Uma sequência de caracteres que especifica o tipo de programa de armazenamento de dados a ser utilizado. Os valores são: STUB VSAM
preenchimento	PIC X	
DO-OUTBOUND-WS	PIC X	Um caractere que especifica se o gerenciador de despacho fez uma solicitação de serviço da web de saída. Os valores são: Y N
preenchimento	PIC X	
CATMAN-PROG	PIC X(8)	O nome do programa do gerenciador de catálogo.
preenchimento	PIC X	
DSSTUB-PROG	PIC X(8)	O nome do programa manipulador de dados simulado.
preenchimento	PIC X	
DSVSAM-PROG	PIC X(8)	O nome do programa manipulador de dados VSAM.
preenchimento	PIC X	
ODSTUB-PROG	PIC X(8)	O nome do módulo de dispatcher do pedido simulado.
preenchimento	PIC X	
ODWEBS-PROG	PIC X(8)	O nome do programa de dispatcher do pedido do serviço da web de saída.
preenchimento	PIC X	
STKMAN-PROG	PIC X(8)	O nome do programa gerenciador de estoque.
preenchimento	PIC X(10)	

Tabela 28. Registro de URL de Saída

Nome	Tipo de Dados COBOL	Descrição
URL-KEY	PIC X(9)	Campo-chave para o registro de informações gerais, contendo OUTBNDURL.
preenchimento	PIC X	

Tabela 28. Registro de URL de Saída (continuação)

Nome	Tipo de Dados COBOL	Descrição
OUTBOUND-URL	PIC X(255)	URL de saída para a solicitação de serviço da web do dispatcher do pedido.

Tabela 29. Registro de Informações do Arquivo de Catálogo

Nome	Tipo de Dados COBOL	Descrição
URL-FILE-KEY	PIC X(9)	Campo-chave para o registro de informações gerais, contendo VSAM-NAME.
preenchimento	PIC X	
CATALOG-FILE-NAME	PIC X(8)	Nome do recurso CICS FILE usado para o arquivo de catálogo.

Tabela 30. Registro de Informações do Servidor

Nome	Tipo de Dados COBOL	Descrição
WS-SERVER-KEY	PIC X(9)	Campo-chave para o registro de informações do servidor, contendo WS-SERVER.
preenchimento	PIC X	
CATALOG-FILE-NAME	PIC X(8)	Apenas para o cliente de serviço da web do CICS, o endereço IP e a porta do sistema no qual o aplicativo de exemplo é implementado como um serviço da web.

Amostras JSON

Utilize estes exemplos para ajudar a entender as solicitações de JSON.

Solicitação HTTP GET de exemplo usando uma sequência de consultas

Este é um exemplo de uma solicitação HTTP GET usando uma sequência de consultas.

```
GET /genapp/customers?name=Joe%20Bloggs/ 1
Host: www.example.com
```

Em que **1** é a sequência de consultas.

Exemplo de solicitação de HTTP com um corpo JSON

Este é um exemplo de uma solicitação de HTTP com um corpo JSON.

```
POST /genapp/customers/
Host: www.example.com
Content-Type: application/json
Content-Length: nn 1
```

```
{
  "customers":
  {
    "firstName": "Joe",
    "lastName": "Bloggs",
    "fullAddress":
```



```
{
  "streetAddress": "21 2nd Street",
  "city": "New York",
  "state": "NY",
  "postalCode": 10021
}
```

Em que o Content-Length: *nn* **1** é o comprimento de sua solicitação.

O mapeamento da estrutura de linguagem COBOL para este exemplo seria conforme a seguir:

```
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).
```

Avisos

Estas informações foram desenvolvidas para produtos e serviços oferecidos nos Estados Unidos. Este material pode estar disponível por meio da IBM em outros idiomas. No entanto, talvez seja necessário ter uma cópia do produto ou da versão do produto naquele idioma para poder acessá-lo.

É possível que a IBM não forneça os produtos, serviços ou recursos discutidos nesta publicação em outros países. Consulte um representante IBM local para obter informações sobre produtos e serviços disponíveis atualmente em sua área. Qualquer referência a produtos, programas ou serviços IBM não significa que apenas produtos, programas ou serviços IBM possam ser utilizados. Qualquer produto, programa ou serviço funcionalmente equivalente, que não infrinja nenhum direito de propriedade intelectual da IBM, poderá ser usado. No entanto, é responsabilidade do usuário avaliar e verificar a operação de qualquer produto, programa ou serviço não IBM.

A IBM pode ter patentes ou solicitações de patentes pendentes relativas a assuntos tratados neste documento. O fornecimento desse documento não concede ao Cliente nenhuma licença para essas patentes. Pedidos de licença devem ser enviados, por escrito, para:

Gerência de Relações Comerciais e Industriais da IBM Brasil
Av. Pasteur, 138-146
Botafogo
Rio de Janeiro, RJ
CEP 22290-240

Para pedidos de licença relacionados a informações de DBCS (Conjunto de Caracteres de Byte Duplo), entre em contato com o Departamento de Propriedade Intelectual da IBM em seu país ou envie pedidos de licença, por escrito, para:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

A INTERNATIONAL BUSINESS MACHINES CORPORATION FORNECE ESTA PUBLICAÇÃO “NO ESTADO EM QUE SE ENCONTRA”, SEM GARANTIA DE NENHUM TIPO, SEJA EXPRESSA OU IMPLÍCITA, INCLUINDO, MAS A ELAS NÃO SE LIMITANDO, AS GARANTIAS IMPLÍCITAS DE NÃO INFRAÇÃO, COMERCIALIZAÇÃO OU ADEQUAÇÃO A UM DETERMINADO PROPÓSITO. Algumas jurisdições não permitem a exclusão de garantias expressas ou implícitas em certas transações; portanto, essa disposição pode não se aplicar ao Cliente.

Estas informações podem conter imprecisões técnicas ou erros tipográficos. São feitas alterações periódicas nas informações aqui contidas; tais alterações serão incorporadas em futuras edições desta publicação. A IBM pode, a qualquer momento, aperfeiçoar e/ou alterar os produtos e/ou programas descritos nesta publicação, sem aviso prévio.

Quaisquer referências nestas informações a websites não IBM são fornecidas apenas para conveniência e não representam de forma alguma um endosso a esses websites. Os materiais contidos nesses websites não fazem parte dos materiais deste produto IBM e o uso desses websites é de inteira responsabilidade do cliente.

A IBM pode utilizar ou distribuir as informações fornecidas da forma que julgar apropriada sem incorrer em qualquer obrigação para com o Cliente.

Licenciados deste programa que desejam obter informações sobre este assunto com objetivo de permitir: (i) a troca de informações entre programas criados independentemente e outros programas (incluindo este) e (ii) a utilização mútua das informações trocadas, devem entrar em contato com

Gerência de Relações Comerciais e Industriais da IBM Brasil
Av. Pasteur, 138-146
Botafogo
Rio de Janeiro, RJ
CEP 22290-240

Tais informações podem estar disponíveis, sujeitas a termos e condições apropriadas, incluindo em alguns casos o pagamento de uma taxa.

O programa licenciado descrito nesta publicação e todo o material licenciado disponível são fornecidos pela IBM sob os termos do Contrato com o Cliente IBM IBM, do Contrato Internacional de Licença do Programa IBM ou de qualquer outro contrato equivalente.

As informações relativas a produtos não IBM foram obtidas junto aos fornecedores dos respectivos produtos, de seus anúncios publicados ou de outras fontes disponíveis publicamente. A IBM não testou esses produtos e não pode confirmar a precisão de seu desempenho, compatibilidade nem quaisquer outras reivindicações relacionadas a produtos não IBM. Dúvidas sobre os recursos de produtos não IBM devem ser encaminhadas diretamente a seus fornecedores.

Estas informações contêm exemplos de dados e relatórios utilizados nas operações diárias de negócios. Para ilustrá-las da forma mais completa possível, os exemplos incluem os nomes de pessoas, empresas, marcas e produtos. Todos estes nomes são fictícios e qualquer semelhança com nomes e endereços usados por uma empresa real é mera coincidência.

COPYRIGHT LICENSE:

Estas informações contêm programas aplicativos de amostra no idioma de origem, que ilustram as técnicas de programação em diversas plataformas operacionais. O Cliente pode copiar, modificar e distribuir estes programas de amostra sem a necessidade de pagar à IBM, com objetivos de desenvolvimento, utilização, marketing ou distribuição de programas aplicativos em conformidade com a interface de programação de aplicativo para a plataforma operacional para a qual os programas de amostra são criados. Esses exemplos não foram totalmente testados sob todas as condições. Portanto, a IBM não pode garantir ou implicar a confiabilidade, manutenção ou função destes programas. Os programas de exemplo são fornecidos "NO ESTADO EM QUE SE ENCONTRAM", sem garantia de nenhum tipo. A IBM não é responsável por nenhum dano decorrente do uso dos programas de amostra.

Informações sobre a interface de programação

O CICS fornece algumas documentações que podem ser consideradas como Interfaces de Programação e alguma documentação que não pode ser considerada como uma Interface de Programação.

As Interfaces de Programação que permitem ao cliente gravar programas para obter os serviços do CICS Transaction Server para z/OS, Versão 5 Release 4 estão incluídas nas seguintes seções da documentação do produto on-line:

- Desenvolvendo Aplicativos
- Desenvolvendo programas do sistema
- Visão geral da segurança do RACF
- Desenvolvendo para interfaces externas
- Referência: desenvolvimento de aplicativos
- Referência: programação do sistema
- Referência: conectividade

Informações que NÃO são destinadas ao uso como uma Interface de Programação do CICS Transaction Server para z/OS, Versão 5 Release 4, mas que podem ser interpretadas erroneamente como Interfaces de Programação, estão incluídas nas seções a seguir da documentação do produto on-line:

- Resoluções de Problemas e Suporte
- Referência: diagnósticos

Se você acessar a documentação do CICS em manuais no formato PDF, as Interfaces de Programação que permitem ao cliente gravar programas para obter os serviços do CICS Transaction Server para z/OS, Versão 5 Release 4 estão incluídas nos seguintes manuais:

- Guia de Programação de Aplicativos e Referência de Programação de Aplicativos
- Serviços de Transação de Negócios
- Guia de Customização
- Bibliotecas de Classe C++ OO
- Referência de Interfaces de Ferramentas de Depuração
- Guia de Programação de Transação Distribuída
- Guia de Interfaces Externas
- Guia de Interface de Programação de Front End
- Guia do IMS Database Control
- Guia de Instalação
- Guia de Segurança
- Transações Fornecidas
- CICSplex SM - Gerenciando Cargas de Trabalho
- CICSplex SM - Gerenciando o Uso de Recurso
- Guia de Programação de Aplicativos e Referência de Programação de Aplicativos do CICSplex SM
- Aplicativos Java no CICS

Se você acessar a documentação do CICS em manuais no formato PDF, as informações que NÃO são destinadas ao uso como uma Interface de Programação

do CICS Transaction Server para z/OS, Versão 5 Release 4 , mas que podem ser interpretadas erroneamente como Interfaces de Programação, estão incluídas nos manuais a seguir:

- Áreas de Dados
- Referência de Diagnóstico
- Guia para Determinação de Problemas
- CICSplex SM Problem Determination Guide

Marcas Registradas

IBM, o logotipo IBM e `ibm.com` são marcas comerciais ou marcas registradas da International Business Machines Corp., registradas em muitos países no mundo todo. Outros nomes de produtos e serviços podem ser marcas comerciais da IBM ou de outras empresas. Uma lista atual de marcas comerciais da IBM está disponível na web em Informações de Copyright e Marcas Comerciais, no endereço www.ibm.com/legal/copytrade.shtml.

Adobe, o logotipo Adobe, PostScript e o logotipo PostScript são marcas ou marcas registradas da Adobe Systems Incorporated nos Estados Unidos e/ou em outros países.

Intel, o logotipo Intel, Intel Inside, o logotipo Intel Inside, Intel Centrino, o logotipo Intel Centrino, Celeron, Intel Xeon, Intel SpeedStep, Itanium e Pentium são marcas ou marcas registradas da Intel Corporation ou de suas subsidiárias nos Estados Unidos e/ou em outros países.

Java e todas as marcas registradas e logotipos baseados em Java são marcas ou marcas registradas da Oracle e/ou suas afiliadas.

Linux é uma marca registrada de Linus Torvalds nos Estados Unidos e/ou em outros países.

Microsoft, Windows, Windows NT e o logotipo Windows são marcas registradas da Microsoft Corporation nos Estados Unidos e/ou em outros países.

UNIX é uma marca registrada do The Open Group nos Estados Unidos e em outros países.

Termos e condições para a documentação do produto

As permissões para uso destes documentos são concedidas de acordo com os termos e condições a seguir.

Aplicabilidade

Esses termos e condições são adicionais a quaisquer termos de uso para o website da IBM.

utilizar o Personal

O Cliente pode reproduzir essas publicações para seu uso pessoal, não comercial, desde que todos os avisos do proprietário sejam preservados. Você não pode distribuir, exibir ou fazer trabalho derivativo destas publicações ou qualquer parte delas, sem o consentimento expresso da IBM.

Uso comercial

É possível reproduzir, distribuir e exibir estas publicações somente dentro

de sua empresa, desde que todos os avisos do proprietário sejam preservados. Não é possível fazer trabalhos derivativos dessas publicações, nem reproduzir, distribuir ou exibir essas publicações, ou de qualquer parte delas fora de sua empresa, sem o consentimento expresso da IBM.

Direitos

Exceto quando concedido expressamente nesta permissão, nenhuma outra permissão, licença ou direito são concedidos, seja de maneira expressa ou implícita, para as publicações ou quaisquer informações, dados, software ou outra propriedade intelectual contida aqui.

A IBM reserva-se o direito de retirar as permissões concedidas aqui, sempre que, a seu critério, o uso das publicações for prejudicial ao seu interesse ou, conforme determinado pela IBM, as instruções anteriores que não estiverem sendo seguidas adequadamente.

Não é permitido fazer download, exportar ou exportar novamente estas informações, exceto em conformidade total com todos os regulamentos e leis aplicáveis, incluindo todos os regulamentos e leis de exportação dos Estados Unidos.

A IBM NÃO OFERECE NENHUMA GARANTIA QUANTO AO CONTEÚDO DESTAS PUBLICAÇÕES. ESTAS PUBLICAÇÕES SÃO FORNECIDAS "NO ESTADO EM QUE SE ENCONTRAM" E SEM QUAISQUER GARANTIAS DE QUALQUER TIPO, EXPRESSAS OU IMPLÍCITAS, INCLUINDO, MAS NÃO SE LIMITANDO A, GARANTIAS IMPLÍCITAS DE COMERCIALIZAÇÃO, NÃO INFRAÇÃO E ADEQUAÇÃO A UM DETERMINADO PROPÓSITO SÃO EXCLUÍDAS.

Declaração de privacidade on-line da IBM

Os produtos de Software IBM, incluindo soluções de software como serviço ("Ofertas de Software"), podem usar cookies ou outras tecnologias para coletar informações de uso do produto, para ajudar a melhorar a experiência do usuário final, para customizar interações com o usuário final ou para outros propósitos. Em muitos casos, nenhuma informação de identificação pessoal é coletada pelas Ofertas de Software. Algumas das Ofertas de Software podem ajudar a permitir a coleta de informações de identificação pessoal. Se esta Oferta de Software usar cookies para coletar informações pessoalmente identificáveis, informações específicas sobre o uso de cookies desta oferta serão descritas abaixo:

Para a Interface com o Usuário da Web do CICSplex SM (interface principal):

Dependendo das configurações implementadas, esta Oferta de Software pode usar cookies de sessão e persistentes que coletam o nome de usuário de cada usuário e outras informações pessoalmente identificáveis para propósitos de gerenciamento de sessão, autenticação, usabilidade do usuário aprimorada ou outros propósitos de rastreamento de uso ou funcionais. Esses cookies não podem ser desativados.

Para a Interface com o Usuário da Web do CICSplex SM (interface de dados):

Dependendo das configurações implementadas, esta Oferta de Software pode usar cookies de sessão que coletam o nome de usuário de cada usuário e outras informações pessoalmente identificáveis para propósitos de gerenciamento de sessão, autenticação ou outros propósitos de rastreamento de uso ou funcionais. Esses cookies não podem ser desativados.

Para o CICSplex SM Web User Interface (página do "hello world"):

Dependendo das configurações implementadas, esta Oferta de Software

pode usar cookies de sessão que não coletam informações pessoalmente identificáveis. Esses cookies não podem ser desativados.

Para o CICS Explorer:

Dependendo das configurações implementadas, esta Oferta de Software pode usar preferências de sessão e persistentes que coletam o nome do usuário e a senha de cada usuário, para os propósitos de gerenciamento de sessões, autenticação e configuração de conexão única. Essas preferências não podem ser desativadas, embora o armazenamento de uma senha de usuário em disco em formato criptografado somente possa ser ativado pela ação explícita do usuário ao marcar uma caixa de seleção durante a conexão.

Se as configurações implementadas para esta Oferta de Software fornecerem a você, como cliente, a capacidade de coletar informações pessoalmente identificáveis de usuários finais por meio de cookies e outras tecnologias, o Cliente deverá buscar seu próprio conselho jurídico sobre as leis aplicáveis a essa coleta de dados, incluindo quaisquer requisitos de aviso e consentimento.

Para obter mais informações sobre o uso de várias tecnologias, incluindo cookies, para esses propósitos, consulte Política de privacidade IBM e Declaração de privacidade on-line da IBM, a seção intitulada “Cookies, Web Beacons e Outras Tecnologias” e o Declaração de Privacidade de Produtos de Software e Software como Serviço IBM.

Índice Remissivo

Caracteres Especiais

<addressing>
 elemento de configuração de enfileiramento 96

<apphandler_class>
 elemento de configuração de enfileiramento 91

<apphandler>
 elemento de configuração de enfileiramento 92

<auth_token_type>
 elemento de configuração de enfileiramento 129

<authentication>
 elemento de configuração de enfileiramento 121

<cics_json_handler_java>
 elemento de configuração de enfileiramento 97

<cics_mtom_handler>
 elemento de configuração de enfileiramento 114

<cics_soap_1.1_handler>
 elemento de configuração de enfileiramento 97

<cics_soap_1.1_handler_java>
 elemento de configuração de enfileiramento 99

<cics_soap_1.2_handler>
 elemento de configuração de enfileiramento 102

<cics_soap_1.2_handler_java>
 elemento de configuração de enfileiramento 104

<default_http_transport_handler_list>
 elemento de configuração de enfileiramento 107

<default_mq_transport_handler_list>
 elemento de configuração de enfileiramento 107

<default_target>
 elemento de configuração de enfileiramento 112

<default_transport_handler_list>
 elemento de configuração de enfileiramento 108

<dfhmtom_configuration>
 elemento de configuração de enfileiramento 115

<dfhwsse_configuration>
 elemento de configuração de enfileiramento 119

<encrypt_body>
 elemento de configuração de enfileiramento 131

<handler>
 elemento de configuração de enfileiramento 108

<jvmserver>
 elemento de configuração de enfileiramento 109

<mime_options>
 elemento de configuração de enfileiramento 118

<mtom>
 elemento de configuração de enfileiramento 113

<mtom_options>
 elemento de configuração de enfileiramento 115

<named_transport_entry>
 elemento de configuração de enfileiramento 92

<namespace>
 elemento de configuração de enfileiramento 96

<provider_pipeline>
 elemento de configuração de enfileiramento 92

<provider_pipeline_json>
 elemento de configuração de enfileiramento 93

<repository>
 elemento de configuração de enfileiramento 110

<requester_pipeline>
 elemento de configuração de enfileiramento 95

<service>
 elemento de configuração de enfileiramento 110

<service_handler_list>
 elemento de configuração de enfileiramento 111

<service_parameter_list>
 elemento de configuração de enfileiramento 112

<sign_body>
 elemento de configuração de enfileiramento 131

<sts_authentication>
 elemento de configuração de enfileiramento 126

<sts_endpoint>
 elemento de configuração de enfileiramento 130

<terminal_handler>
 elemento de configuração de enfileiramento 94

<transport>
 elemento de configuração de enfileiramento 112

<transport_handler_list>
 elemento de configuração de enfileiramento 95

<wsse_handler>
 elemento de configuração de enfileiramento 119

<xop_options>
 elemento de configuração de enfileiramento 117

A

algorithm (algoritmo) 610, 612

Amostra 643

Amostra de Catálogo amostra de catálogo 643

Amostras JSON 688

análise de XML 398

anexo binário
 configuração de pipeline 113

aplicativo
 transformar em XML 289

aplicativo do provedor de serviços 272
 criando a partir de uma estrutura de dados 274, 577
 usando transações atômicas 195

aplicativo do solicitante de serviço
 usando transações atômicas 196

apphandler
 elemento de configuração de enfileiramento 92

arquivo de configuração, pipeline 79

arquivo de configuração de pipeline 79

arquivo de ligação
 ligação XML 402

assistente, JSON 234

Assistente, JSON 409

assistente, serviços da web 487

assistente, XML 290

assistente de serviços da web 487
 criando um aplicativo do provedor de serviços 274, 577

assistente JSON 234, 409, 638, 641

assistente XML 290
 DFHLS2SC 402

autenticação
 elemento de configuração de enfileiramento 121

auth_token_type
 elemento de configuração de enfileiramento 129

Axis2 55

C

C and C++
 mapeamento para Esquema XML 327, 330
 mapeando para Esquema JSON 449, 452, 543, 546

cabeçalho, SOAP 20

caminho da mensagem SOAP 8

chamando o cliente de confiança 623

chamar serviço da web a partir de Java 596

cics_json_handler_java
 elemento de configuração de
 enfileiramento 97

cics_mtom_handler
 elemento de configuração de
 enfileiramento 114

cics_soap_1.1_handler
 elemento de configuração de
 enfileiramento 97

cics_soap_1.1_handler_java
 elemento de configuração de
 enfileiramento 99

cics_soap_1.2_handler
 elemento de configuração de
 enfileiramento 102

cics_soap_1.2_handler_java
 elemento de configuração de
 enfileiramento 104

cliente de confiança
 chamando 623
 externa 609

COBOL
 conteúdo de repetição variável 360,
 390
 mapeamento para esquema XML 320
 mapeamento para Esquema XML 311
 mapeando para esquema JSON 442,
 536
 mapeando para Esquema JSON 434,
 528

código de retorno 641

comando EXEC CICS SOAPFAULT
 CREATE 584

configuração 74

configuração de pipeline
 MTOM/XOP 113
 Segurança de Serviços da Web 118

configurando o pipeline 618

configurando o RACF 614

consultando XML 397

contêiner
 contêiner de contextos
 DFH-HANDLERPLIST 160
 DFH-SERVICEPLIST 160
 DFHWS-APPHANDLER 160, 161,
 163
 DFHWS-DATA 161
 DFHWS-FAULT 162
 DFHWS-PIPELINE 163
 DFHWS-SOAPLEVEL 164
 DFHWS-STSREASON 176
 DFHWS-TRANID 164
 DFHWS-URI 164
 DFHWS-USERID 168
 DFHWS-WEBSERVICE 169
 contêiner de controle
 DFHERROR 150
 DFHFHFUNCTION 152
 DFHHTTPSTATUS 154
 DFHMEDIATYPE 155
 DFHNORESPONSE 155
 DFHREQUEST 155
 DFHRESPONSE 156
 DFHWS-CCSID 156
 DFH-EXIT-HEADER1 160
 DFH-HANDLERPLIST 160
 DFH-SERVICEPLIST 160
 contêiner (continuação)
 DFHERROR 150
 DFHFHFUNCTION 152
 DFHHTTPSTATUS 154
 DFHMEDIATYPE 155
 DFHNORESPONSE 155
 DFHREQUEST 155
 DFHRESPONSE 156
 DFHWS-APPHANDLER 160, 161,
 163
 DFHWS-CCSID 156
 DFHWS-CID-DOMAIN 169
 DFHWS-DATA 161
 DFHWS-FAULT 162
 DFHWS-IDTOKEN 174
 DFHWS-LOCATION 162
 DFHWS-MEP 162
 DFHWS-MTOM-IN 169
 DFHWS-MTOM-OUT 170
 DFHWS-PIPELINE 163
 DFHWS-RESPWAIT 163
 DFHWS-RESTOKEN 175
 DFHWS-SERVICEURI 175
 DFHWS-SOAPLEVEL 164
 DFHWS-STSACTION 175
 DFHWS-STSFAULT 175
 DFHWS-STSREASON 176
 DFHWS-STSURI 176
 DFHWS-TOKENTYPE 176
 DFHWS-TRANID 164
 DFHWS-URI 164
 DFHWS-USERID 168
 DFHWS-WEBSERVICE 169
 DFHWS-XOP-IN 172
 DFHWS-XOP-OUT 171, 172
 contêiner de contextos
 DFH-EXIT-HEADER1 160
 DFHWS-CID-DOMAIN 169
 DFHWS-IDTOKEN 174
 DFHWS-LOCATION 162
 DFHWS-MEP 162
 DFHWS-MTOM-IN 169
 DFHWS-MTOM-OUT 170
 DFHWS-RESPWAIT 163
 DFHWS-RESTOKEN 175
 DFHWS-SERVICEURI 175
 DFHWS-STSACTION 175
 DFHWS-STSFAULT 175
 DFHWS-STSURI 176
 DFHWS-TOKENTYPE 176
 DFHWS-WSDL-CTX 171
 DFHWS-XOP-IN 172
 DFHWS-XOP-OUT 172
 Contêiner DFH-EXIT-HEADER1 160
 Contêiner DFH-HANDLERPLIST 160
 Contêiner DFH-SERVICEPLIST 160
 Contêiner DFHERROR 150
 Contêiner DFHFHFUNCTION 152
 Contêiner DFHHTTPSTATUS 154
 Contêiner DFHMEDIATYPE 155
 Contêiner DFHNORESPONSE 155
 Contêiner DFHREQUEST 155
 Contêiner DFHRESPONSE 156
 contêiner DFHSAML-AnnnVmmm 177
 contêiner DFHSAML-ASSQNAME 177
 contêiner DFHSAML-ATTRAnnn 177
 contêiner DFHSAML-ATTRFnnn 177
 contêiner DFHSAML-ATTRNnnn 177
 contêiner DFHSAML-CERTIDN 178
 contêiner DFHSAML-CERTSDN 178
 contêiner DFHSAML-CERTSNUM 178
 contêiner DFHSAML-CONFMETHOD 178
 contêiner DFHSAML-COUNTS 179
 contêiner DFHSAML-FLAGS 179
 contêiner DFHSAML-ISSUER 179
 contêiner DFHSAML-NAMID 179
 contêiner DFHSAML-NAMIDF 179
 contêiner DFHSAML-NAMIDQ 179
 contêiner DFHSAML-NAMIDSP 179
 contêiner DFHSAML-NAMIDSPQ 179
 contêiner DFHSAML-OUTTOKEN 179
 contêiner DFHSAML-PROXYnnn 180
 contêiner DFHSAML-RESPONSE 180
 contêiner DFHSAML-SAMLID 180
 contêiner DFHSAML-SUBJADDR 180
 contêiner DFHSAML-SUBJDNS 180
 contêiner DFHSAML-TIMES 180
 Contêiner DFHWS-APPHANDLER 160,
 161, 163
 Contêiner DFHWS-CCSID 156
 Contêiner DFHWS-CID-DOMAIN 169
 Contêiner DFHWS-DATA 161
 Contêiner DFHWS-FAULT 162
 Contêiner DFHWS-IDTOKEN 174
 Contêiner DFHWS-LOCATION 162
 Contêiner DFHWS-MEP 162
 Contêiner DFHWS-MTOM-IN 169
 Contêiner DFHWS-MTOM-OUT 170
 Contêiner DFHWS-PIPELINE 163
 Contêiner DFHWS-RESPWAIT 163
 Contêiner DFHWS-RESTOKEN 175
 Contêiner DFHWS-SERVICEURI 175
 Contêiner DFHWS-SOAPLEVEL 164
 Contêiner DFHWS-STSACTION 175
 Contêiner DFHWS-STSFAULT 175
 Contêiner DFHWS-STSREASON 176
 Contêiner DFHWS-STSURI 176
 Contêiner DFHWS-TOKENTYPE 176
 Contêiner DFHWS-TRANID 164
 Contêiner DFHWS-URI 164
 Contêiner DFHWS-USERID 168
 Contêiner DFHWS-WEBSERVICE 169
 Contêiner DFHWS-WSDL-CTX 171
 Contêiner DFHWS-XOP-IN 172
 Contêiner DFHWS-XOP-OUT 172
 contêineres
 descrição de canal 276, 580
 DFHSAML-AnnnVmmm 177
 DFHSAML-ASSQNAME 177
 DFHSAML-ATTRAnnn 177
 DFHSAML-ATTRFnnn 177
 DFHSAML-ATTRNnnn 177
 DFHSAML-ATTRSnnn 177
 DFHSAML-ATTRYnnn 178
 DFHSAML-AUDNRnnn 178
 DFHSAML-AUTHMETH 178
 DFHSAML-CERTIDN 178
 DFHSAML-CERTSDN 178
 DFHSAML-CERTSNUM 178
 DFHSAML-CONFMETHOD 178

contêineres (*continuação*)
 DFHSAML-COUNTS 179
 DFHSAML-FLAGS 179
 DFHSAML-ISSUER 179
 DFHSAML-NAMID 179
 DFHSAML-NAMIDF 179
 DFHSAML-NAMIDQ 179
 DFHSAML-NAMIDSP 179
 DFHSAML-NAMIDSPQ 179
 DFHSAML-OUTTOKEN 179
 DFHSAML-PROXYnnn 180
 DFHSAML-RESPONSE 180
 DFHSAML-SAMLID 180
 DFHSAML-SUBJADDR 180
 DFHSAML-SUBJDNS 180
 DFHSAML-TIMES 180
 SAML 176

contêineres de contextos 159
 contêineres de controle 150
 contêineres de segurança 174
 DFHSAML-AnnnVmmm 177
 DFHSAML-ASSQNAME 177
 DFHSAML-ATTRAnnn 177
 DFHSAML-ATTRFnnn 177
 DFHSAML-ATTRNnnn 177
 DFHSAML-ATTRSnnn 177
 DFHSAML-ATTRYnnn 178
 DFHSAML-AUDNRnnn 178
 DFHSAML-AUTHMETH 178
 DFHSAML-CERTIDN 178
 DFHSAML-CERTSDN 178
 DFHSAML-CERTSNUM 178
 DFHSAML-CONFMETH 178
 DFHSAML-COUNTS 179
 DFHSAML-FLAGS 179
 DFHSAML-ISSUER 179
 DFHSAML-NAMID 179
 DFHSAML-NAMIDF 179
 DFHSAML-NAMIDQ 179
 DFHSAML-NAMIDSP 179
 DFHSAML-NAMIDSPQ 179
 DFHSAML-OUTTOKEN 179
 DFHSAML-PROXYnnn 180
 DFHSAML-RESPONSE 180
 DFHSAML-SAMLID 180
 DFHSAML-SUBJADDR 180
 DFHSAML-SUBJDNS 180
 DFHSAML-TIMES 180

contêineres do usuário 181
 conteúdo de repetição 360, 390
 corpo, SOAP 20
 Criando um aplicativo do solicitante de
 serviço 485
 customizando o processamento de
 pipeline 187

D

default_http_transport_handler_list
 elemento de configuração de
 enfileiramento 107, 109, 110
 default_mq_transport_handler_list
 elemento de configuração de
 enfileiramento 107
 default_target
 elemento de configuração de
 enfileiramento 112

default_transport_handler_list
 elemento de configuração de
 enfileiramento 108
 definição de pipeline
 solicitante de serviço 89
 descoberta de serviços da web 43
 descrição de canal 276, 580
 determinação de problemas 637
 DFHJS2LS
 procedimento catalogado 246, 259,
 418
 DFHLS2JS
 procedimento catalogado 234, 410
 DFHLS2SC
 procedimento catalogado 290
 DFHLS2SC, programa assistente XML do
 CICS 402
 DFHLS2WS
 procedimento catalogado 488
 dfhmtom_configuration
 elemento de configuração de
 enfileiramento 115
 DFHSC2LS
 procedimento catalogado 298
 DFHWS2LS
 procedimento catalogado 504
 dfhwse_configuration
 elemento de configuração de
 enfileiramento 119
 diagnosticando problemas
 fornecedor de serviços 630
 solicitante de serviços 631
 documento WSDL
 comprimento variável 368, 393, 569
 espaço em branco 368, 393, 569

E

elemento de configuração de
 enfileiramento
 <addressing> 96
 <apphandler> 92
 <auth_token_type> 129
 <authentication> 121
 <cics_json_handler_java> 97
 <cics_mtom_handler> 114
 <cics_soap_1.1_handler> 97
 <cics_soap_1.1_handler_java> 99
 <cics_soap_1.2_handler> 102
 <cics_soap_1.2_handler_java> 104
 <default_http_transport_
 handler_list> 107
 <default_mq_transport_
 handler_list> 107
 <default_transport_handler_list> 108
 <dfhmtom_configuration> 115
 <dfhwse_configuration> 119
 <encrypt_body> 131
 <handler> 108
 <jvmserver> 109
 <mime_options> 118
 <mtom> 113
 <mtom_options> 115
 <named_transport_entry> 92
 <namespace> 96
 <provider_pipeline> 92
 <provider_pipeline_json> 93

elemento de configuração de
 enfileiramento (*continuação*)
 <repository> 110
 <requester_pipeline> 95
 <service> 110
 <service_handler_list> 111
 <sign_body> 131
 <sts_authentication> 126
 <sts_endpoint> 130
 <terminal_handler> 94
 <transport> 112
 <transport_handler_list> 95
 <wsse_handler> 119
 <xop_options> 117
 encrypt_body
 elemento de configuração de
 enfileiramento 131
 endereçamento
 elemento de configuração de
 enfileiramento 96
 Endereço WS
 <wsa:Action> 219
 ações explícitas 219
 ações padrão 220, 221
 configuração do pipeline do
 provedor 215
 configuração do pipeline do
 solicitante 213
 DFHWSADH 213, 215
 EPR padrão 218
 Especificação 209
 manipulador de endereçamento 213,
 215
 parâmetros 520
 WSADDR-EPR-ANY 520
 WSDL 1.1 220
 WSDL 2.0 221
 envelope, SOAP 20
 EPR padrão 218
 WSDL 1.1 218
 Erro 641
 erros de serviço da web 630, 631
 espaço de nomes
 elemento de configuração de
 enfileiramento 96
 esquema
 descrição de canal 276, 580
 esquema JSON 272, 442, 536
 Esquema JSON 434, 449, 452, 458, 464,
 528, 543, 546, 552, 558
 convertendo para JSON 410
 esquema XML
 comprimento variável 368, 393, 569
 convertendo em WSDL 290
 espaço em branco 368, 393, 569
 Esquema XML 311, 320, 327, 330, 335,
 340
 estrutura de linguagem
 convertendo em WSDL 290, 488
 convertendo para JSON 234, 410
 estrutura de linguagem de alto nível
 convertendo em WSDL 488
 convertendo para JSON 234
 exemplo 643

F

falha, SOAP 20
falhas de SOAP 584
feed Atom
 ligação XML
 criando 402
fornecedor de serviços
 diagnosticando problemas 630

G

Gerando mapeamentos a partir da
 estrutura de linguagem 478
Gerando mapeamentos a partir de um
 esquema JSON 480
gerenciamento de carga de trabalho
 em um manipulador de terminal 148
 em um provedor de serviços 148
GLUEs 187

H

handler
 elemento de configuração de
 enfileiramento 108

I

instalação 637
instalar 637
interface de canal 276, 580

J

Java 55
 chamar serviço da web 596
JSON 74, 167, 168, 637, 638, 639, 640,
 641, 688
 convertendo em estrutura de
 linguagem 418

L

ligação XML 402
limitações do tempo de execução 184
limitações no tempo de execução 184
linha de comandos 641
linhas de comandos 641
lista de parâmetros de serviço
 <service_parameter_list> 112

M

manipulador de aplicativo
 elemento de configuração de
 enfileiramento 91
manipulador de mensagem não do
 terminal 138, 139, 140
manipulador de mensagens
 chamando o cliente de confiança 623
 não terminal 138, 139, 140
manipulador de segurança
 gravando seu próprio 622

manipulador de segurança
 customizada 622
mapeamento ao PL/I 335, 458, 464, 552,
 558
mapeamento para C e C++ 327, 330,
 449, 452, 543, 546
mapeamento para COBOL 311, 320, 434,
 442, 528, 536
mapeamento para PL/I 340
mapeando esquemas XML para dados do
 aplicativo 404
mapeando estruturas de linguagem para
 XML 402
matrizes Variáveis 282, 347, 363, 371,
 376, 471, 565
maxOccurs
 no esquema JSON 282, 371, 471
 no esquema XML 347, 363, 376, 565
mensagem MIME
 configuração de pipeline 113
mensagem persistente 49
mensagem SOAP
 assinando 610
 criptografando 612
 Estrutura 20
 exemplo 20
Mensagens SOAP
 Esquema XML
 validando a mensagem SOAP 597
 validando no Esquema XML 597
MEP 13
MEP (padrão de troca de mensagens) 13
mime_options
 elemento de configuração de
 enfileiramento 118
minOccurs
 no esquema JSON 282, 371, 471
 no esquema XML 347, 363, 376, 565
modo de compatibilidade 201
modo direto 201
mtom
 elemento de configuração de
 enfileiramento 113
mtom_options
 elemento de configuração de
 enfileiramento 115
MTOM/XOP
 configuração de pipeline 113

N

named_transport_entry
 elemento de configuração de
 enfileiramento 92

P

pipelines SOAP 55
Pipelines SOAP Baseados em Java 213
PL/I
 mapeamento para Esquema
 XML 335, 340
 mapeando para Esquema JSON 458,
 464, 552, 558
processamento de pipeline
 customização 187

processamento de pipeline (*continuação*)
 substituindo o URI 189
programa utilitário
 assistente de serviços da web 487
 assistente JSON 234, 409
 assistente XML 290
provider_pipeline
 elemento de configuração de
 enfileiramento 92

R

referência de terminal
 padrão 218
referência de terminal (EPR) 210
requester_pipeline
 elemento de configuração de
 enfileiramento 95
 elemento de definição de pipeline 89
roteamento dinâmico
 em um manipulador de terminal 148
 em um provedor de serviços 148

S

saídas de usuário global 187
SAML
 contêineres 176
 script dinâmico 637
Security Token Service
 interface com o cliente de
 confiança 609
Segurança de serviços da web
 configuração de pipeline 118
Segurança de Serviços da Web
 (WSS) 603, 614, 618
segurança para serviços da web 603
service_handler_list
 elemento de configuração de
 enfileiramento 111
service_parameter_list
 lista de parâmetros de serviço 112
serviços
 elemento de configuração de
 enfileiramento 110
Serviços da Web do CICS
 Unicode 395, 476, 572
 UTF-16 395, 476, 572
servidor JVM 55, 596
sign_body
 elemento de configuração de
 enfileiramento 131
SOAP
 corpo 20
 envelope 20
 falha 20
 header 20
 visão geral 15
 visão geral de SOAP 15
solicitante de serviços
 definição de pipeline 89
 diagnosticando problemas 631
solução de problemas 637, 641
sts_authentication
 elemento de configuração de
 enfileiramento 126

- sts_endpoint
 - elemento de configuração de enfileiramento 130
- substituindo o URI 189
- suporte de mensagem persistente 50

T

- terminal_handler
 - elemento de configuração de enfileiramento 94
- Tipos de dados xsd:any 400
- transação atômica 190, 197
 - configurando o CICS 193
 - configurando o provedor de serviços 195
 - configurando o solicitante de serviço 196
 - estados 198
 - serviços de registro 191
- TRANSFORM DATATOXML 406
- TRANSFORM XMLTODATA 407
- transformando dados binários em XML 406
- Transformando dados do aplicativo em JSON 481
- Transformando JSON em dados do aplicativo 483
- Transformando JSON usando a interface vinculável 408
- transformando XML 398
- transformando XML em dados binários 407
- transport
 - elemento de configuração de enfileiramento 112
- transport_handler_list
 - elemento de configuração de enfileiramento 95

U

- URI
 - para transporte do WebSphere MQ 47
- utilitário em lote 234
 - assistente de serviços da web 487
 - assistente JSON 409
 - assistente XML 290

V

- validando mensagens SOAP 597

W

- Web Services Addressing
 - <wsa:Action> 219
 - ações explícitas 216, 219
 - ações padrão 216, 220, 221
 - configuração do pipeline do provedor 215
 - configuração do pipeline do solicitante 213
 - DFHWS-URI 210

- Web Services Addressing (*continuação*)
 - DFHWSADH 213, 215
 - EPR 210
 - EPR padrão 216, 218
 - Especificação 209
 - manipulador de endereçamento 213, 215
 - MAP 210
 - parâmetros 520
 - serviço do solicitante 216
 - support 209
 - WSADDR-EPR-ANY 520
 - WSDL 1.1 220
 - WSDL 2.0 221
- WS-Addressing
 - DFHWS-URI 210
 - EPR 210
 - MAP 210
- WS-AT 190
- WSDL
 - consultando 43
 - convertendo em estrutura de linguagem 246, 259, 298, 504
 - e a estrutura de dados do aplicativo 11
 - Web Services Addressing 216
- WSDL 1.1
 - EPR padrão 218
- wsse_handler
 - elemento de configuração de enfileiramento 119

X

- XML
 - análise sintática 398
 - assistente 289
 - consultando 397
 - tipos de dados 398
 - Tipos de dados xsd:any 400
 - transformando 398
 - transformar em dados 289
- xop_options
 - elemento de configuração de enfileiramento 117

Z

- z/OS Connect 74, 77
 - configuração 70, 72
 - Segurança 624
 - visão geral 35
- zAAP 55
- zosConnectService 74

